TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2018-2019

Detección de Copia-Pega Intra-Fotograma en Vídeos basada en el Patrón del Ruido del Sensor

JORGE DÍEZ SÁNCHEZ-CABALLERO

Directores:

Luis Javier García Villalba

Ana Lucila Sandoval Orozco

DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

MADRID, SEPTIEMBRE DE 2019

Agradecimientos

En primer lugar, quiero agradecer a mi familia, por haberme permitido estudiar la carrera que siempre había despertado mi interés, por el apoyo incondicional en los momentos difíciles y por inculcarme los valores por los que ahora me rijo.

A mis amigos, por las alegrías y experiencias compartidas, por ofrecerme su ayuda desinteresada y por revivir mi motivación en los momentos en los que era más débil durante todos estos años.

A mis compañeros de clase por hacer más llevadero, más sencillo y completo mi paso por la universidad y mi día a día en las aulas.

A mis compañeros de trabajo, por su compresión, sus ánimos y su preocupación por mi desarrollo laboral, académico y, sobre todo, personal.

Y, para terminar, gracias a Luis Javier García Villalba, Ana Lucila Sandoval Orozco, Esteban Armas Vega, Edgar González Fernández y a todo el grupo de investigación GASS por dirigirme en este trabajo y por hacerlo posible con su paciencia, compresión y dedicación infinitos.

A todos ellos, muchas gracias.

Abstract

We live on a society where almost everyone uses a mobile device capable of recording videos and capturing photos. As a consequence of that, this type of content is often provided as evidence in many judicial proceedings. With the advances on technology, recording videos has given rise to a new problem since not only it becomes easier to obtain this kind of evidences, but also the forgery techniques for unethical purposes have improved. Thus, forensic analysis of digital videos and images has become one of the most important applications of computer science. There are several types of forgeries that can be applied to this kind of material. In this paper there is a proposal for detecting intra-frame Copy-Moves, a technique by which a region of a frame is copied into another position of it, in order to duplicate or hide an element of the scene. In view of the results obtained on testing, the proposed method seems to effectively detect intra-frame Copy-Moves with static background, such as the ones that could be made over security or traffic cameras.

Palabras clave: *Classification, Compression, Digital Videos, Discrete Wavelet Transform (DWT), Forgery Detection, Key frames, Moving Picture Experts Group (MPEG), Photo Response Non-Uniformity (PRNU), Sensor Pattern Noise, Video Forensics Analysis.*

Resumen

Actualmente, vivimos en una sociedad donde casi cualquier persona utiliza en su día a día un dispositivo móvil capaz de capturar fotos y vídeos. Como consecuencia, este tipo de material informático es aportado como evidencia en numerosos procesos judiciales. Con los avances de la tecnología, esto ha dado lugar a un nuevo problema ya que no sólo han aumentado las facilidades para obtener este tipo de pruebas, sino también las técnicas para falsificarlas o manipularlas para cualquier fin poco ético. Por este motivo, en los últimos años el análisis forense de imágenes y vídeos digitales se ha convertido en una de las principales aplicaciones de la informática. Existen numerosos tipos de manipulaciones que se pueden realizar sobre este tipo de material. En este trabajo, se realiza una propuesta de un algoritmo capaz de detectar manipulaciones de tipo Copia-Pega intra-fotograma, una técnica por la cual una región de un fotograma se copia sobre otra posición de este, ya sea con el fin de duplicar u ocultar un elemento de la escena. A la vista de los resultados obtenidos con los experimentos realizados, se puede concluir que el método propuesto consigue detectar de manera eficaz las manipulaciones Copia-Pega intra-fotograma con fondo estático, como las que podrían aplicar sobre imágenes de cámaras de seguridad o de tráfico.

Palabras clave: *Análisis Forense de Vídeos, Clasificación, Compresión, Detección de Manipulaciones, Fotogramas Clave, Moving Picture Experts Group (MPEG), Patrón de ruido del sensor, Ruido de Respuesta No Uniforme (PRNU), Transformada Discreta de Wavelet (DWT), Vídeos Digitales.*

Lista de Acrónimos

BMP	Mapa de Bits
CCD	*Charge Coupled Device.*
CFA	*Color Filter Array*
CRF	Función de Respuesta de la Cámara
Dataset	Conjunto de datos
DCT	Transformada Discreta del Coseno
DWT	Transformada Discreta de Wavelet
EXIF	*Exchangeable Image File Format*
FFT	Transformada de Fourier
FPS	Fotogramas por segundo
GOP	Grupo de Imágenes
HD	Alta Definición
HSV	*Hue-Saturation-Value*
IQM	*Image Quality Metrics*
JPEG	*Joint Photographic Experts Group*
MBM	Modo de Macrobloque
MIT	*Massachusetts Institute of Technology*
MPEG	*Moving Picture Experts Group*
PCA	Análisis de Componentes Principales
PNG	*Portable Network Graphics*
PRNU	Patrón de Ruido de Respuesta no Uniforme
RBF	Función de Base Radial

RGB	Rojo-Verde-Azul
SPN	*Sensor Pattern Noise*
SVM	Máquina de Soporte Vectorial
VM	Vector de Movimiento

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Motivación

La motivación que impulsa este Trabajo Fin de Grado es la investigación, construcción y evaluación de un algoritmo para la detección de manipulaciones, cada vez más frecuentes en una sociedad donde el flujo de información es cada vez mayor, y, sin embargo, la contrastación de la información cada vez menor.

De este modo, el fin último de este trabajo es aprovechar los recursos proporcionados por la universidad, junto con el asesoramiento de mis tutores y mi propio trabajo de investigación, para aumentar mi conocimiento sobre un campo en desarrollo, cada vez más relevante; así como adaptar los conceptos aprendidos estos años a la resolución de problemas y optimización de algoritmos, más concretamente, al utilizado como base.

Para ello se revisará el estado del arte para conocer el estado actual de la investigación y estudiar la posibilidad de introducir mejoras que aumenten la eficiencia de la detección. Los resultados de la detección se visualizarán gráficamente. Todo ello con la finalidad de que este trabajo pueda servir como base para futuros proyectos relacionados con el análisis forense.

1.2. Contexto

El presente Trabajo Fin de Grado se enmarca dentro de un proyecto de investigación titulado RAMSES aprobado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 (Convocatoria H2020-FCT-2015, Acción de Innovación, Número de Propuesta: 700326)y en el que participa el Grupo GASS del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid (Grupo de Análisis, Seguridad y Sistemas,

http://gass.ucm.es, grupo 910623 del catálogo de grupos de investigación reconocidos por la UCM).

Además de la Universidad Complutense de Madrid participan las siguientes entidades:

- Treelogic Telemática y Lógica Racional para la Empresa Europea SL (España)
- Ministério da Justiça (Portugal)
- University of Kent (Reino Unido)
- Centro Ricerche e Studi su Sicurezza e Criminalità (Italia)
- Fachhochschule fur Offentliche Verwaltung und Rechtspflege in Bayern (Alemania)
- Trilateral Research & Consulting LLP (Reino Unido)
- Politecnico di Milano (Italia)
- Service Public Federal Interieur (Bélgica)
- Universitaet des Saarlandes (Alemania)
- Dirección General de Policía - Ministerio del Interior (España)

1.3. Objetivos de la Investigación

Este trabajo tiene los siguientes objetivos:

- Obtener una base de conocimiento para entender el análisis forense y conceptos relacionados con los vídeos digitales como la captura, formación, compresión, eliminación del ruido, etc.

- Realizar un estudio de la literatura actual para clasificar y sintetizar las principales técnicas de manipulación, detallando los aspectos más importantes de cada una de ellas.

- Estudiar el estado del arte de técnicas de detección de las manipulaciones de vídeos haciendo énfasis en las técnicas de detección de Copia- Pega en

imágenes y vídeos digitales.

- Diseñar e implementar un algoritmo capaz de detectar falsificaciones de tipo Copia-Pega intra-fotograma con fondo estático, como las que se podrían realizar sobre vídeos captados por cámaras de seguridad.

- Evaluar los resultados del algoritmo propuesto utilizando diferentes datasets.

- Colaborar en la investigación del análisis forense, sirviendo como fuente o herramienta para futuros trabajos.

1.4. Planificación del Proyecto

La realización del trabajo ha sido dividida en 3 fases para una mayor organización:

1. **Fase de Investigación**: Esta fase fue la más extensa ya que implicaba comprender el contexto en el que se desarrollará el trabajo, analizar el estado actual de las investigaciones relacionadas, encontrar las debilidades de las técnicas y proponer posibles formas de solucionarlas. En esta fase se realizaron las siguientes actividades:

    - La primera fase del proyecto consistió principalmente en leer artículos, estudios y publicaciones científicas, con el objetivo de obtener toda la información posible sobre el tema antes de abordarlo; ya que, aunque el análisis forense en la informática me ha suscitado interés en varias ocasiones, nunca me había planteado desarrollar algo relacionado y, por tanto, el TFG lo inicié desde un nivel muy básico de conocimientos. Se estudiaron, entre otros, documentos sobre los componentes hardware de una cámara, el proceso de generación de imágenes y videos desde que son capturados por un dispositivo móvil, los métodos de compresión de vídeos, motivaciones y usos de la alteración de imágenes y vídeos en el día a día, diferentes manipulaciones que se

pueden realizar sobre estos formatos, posibles medidas para la detección o prevención de dichas manipulaciones, etc.

- A continuación, profundicé especialmente en las técnicas de manipulación de imágenes y vídeos, clasificándolas en distintas categorías y observando y recolectando conjuntos de datos de cada una de ellas. Paralelamente, fui estudiando y comparando algoritmos y propuestas para la detección de las manipulaciones anteriores, entendiendo sus fundamentos matemáticos y estadísticos; hasta elegir uno cuya complejidad, utilidad y tecnología fuesen coherentes con el alcance de este TFG.

- Llegados a este punto, se seleccionaron los algoritmos que servirían como base para el diseño del algoritmo a desarrollar, basados en aquellos que tuvieran mejores resultados analizando el patrón del ruido que deja el sensor de la cámara en los fotogramas del vídeo.

2. **Fase de Ejecución**: Esta fase está compuesta por las siguientes etapas: diseño, implementación y pruebas. En ella se preparó el entorno de desarrollo y se aplicaron los conocimientos adquiridos en el desarrollo de software y preparación de los datos de prueba. Las actividades desarrolladas en esta fase son las siguientes:

   - Basado en esas investigaciones previas se procedió al diseño e implementación de un algoritmo siguiendo la guía de la investigación seleccionada, documentando el proceso y los recursos utilizados en todo momento. Durante esta fase surgieron imprevistos con el algoritmo elegido y hubo que realizar algunos ajustes en la investigación para solucionarlo. En algunas ocasiones tocó volver a la fase anterior, estudiando formas alternativas para solucionar los problemas encontrados.

- Una vez se terminó la implementación, se realizaron pruebas con diversos datasets utilizados en otras investigaciones relacionadas, recopilándolos desde Internet. Es aquí donde se fue afinando poco a poco el algoritmo identificando sus puntos débiles. Se estudiaron diversas modificaciones que mejorarían la tasa de detección y la escalabilidad del algoritmo.

- Una vez implementadas las mejoras al algoritmo se realizó otro grupo de experimentos para comparar los resultados.

3. **Fase de Documentación y Seguimiento**: Esta fase se desarrolló en paralelo con las dos fases anteriores. En ella se documentaba todo lo realizado en la ejecución del TFG y se hacía seguimiento y control del trabajo de investigación en la Fase 1 y soporte técnico en el desarrollo de la Fase 2. En ella se realizaron las siguientes actividades:

    - Se realizaron reuniones semanales de coordinación del trabajo con los directores del proyecto. Inicialmente las reuniones eran dos veces por semana en las que se hacía seguimiento del trabajo, y además se incluían talleres para adquisición de competencias de investigación como: Metodologías de investigación, búsqueda bibliográfica apropiada para la investigación, uso y compilación de documentos en LaTex, herramientas para la gestión de referencias bibliográficas, herramientas de software libre para el análisis de datos como Python, técnicas estadísticas para el procesamiento de señales, entre otras.

    - Todo lo realizado en el proyecto se iba documentando digitalmente para ir preparando en paralelo la memoria del TFG. Una vez finalizadas las pruebas del algoritmo, se sacaron conclusiones sobre los experimentos realizados y se propusieron actividades de trabajo

futuro que puedan desarrollarse para mejorar el algoritmo desarrollado y continuar avanzando la investigación.

1.5. Estructura del Trabajo

El resto de la memoria está organizada en 5 capítulos más con la estructura que se comenta a continuación.

En el capítulo 2 se hace una introducción a los conceptos relacionados con el análisis forense de imágenes y vídeos digitales, ya que son necesarios para el completo entendimiento de este trabajo. Se explicará brevemente el proceso de formación de las imágenes digitales, desde que son capturadas por la cámara del dispositivo móvil hasta que son utilizadas para crear la secuencia del vídeo que será comprimido y almacenado. Además, se explicarán y clasificarán las técnicas de manipulación de vídeos más comunes, acompañadas de ejemplos gráficos.

El capítulo 3 presenta un estado del arte de las principales técnicas de detección de los distintos tipos de manipulaciones, ya comentados en el capítulo anterior, así como las ventajas e inconvenientes de cada uno de los algoritmos analizados. Se hará hincapié en el método que ha motivado y servido como base para desarrollar la técnica propuesta.

El capítulo 4 describe detalladamente el algoritmo de detección de copia-pega desarrollado, explicando el trabajo que se ha realizado y todas las herramientas que han sido utilizadas para lograr el resultado.

En el capítulo 5 muestra los experimentos realizados para evaluar la eficiencia del algoritmo propuesto. Aquí se detallan los datasets que se han utilizado en las pruebas, y se realiza un análisis y comparación de los resultados obtenidos en cada una de ellas.

Finalmente, en el capítulo 6 se exponen las principales conclusiones de este trabajo y una breve reflexión de las posibles líneas de trabajo futuro.

2. Introducción al Análisis Forense en Imágenes y Vídeos Digitales

El análisis forense de imágenes y vídeos digitales es una disciplina que se encarga de la investigación y estudio de evidencias de este formato para determinar incógnitas como su origen, autoría o incluso su autenticidad, que es en lo que se centra en este trabajo.

A lo largo de este capítulo se pretende asentar las bases y conceptos necesarios para poder entender el resto de la memoria. Se explicará el proceso de captura y formación de las imágenes digitales, su método de compresión para la obtención de un vídeo, las principales técnicas de manipulación en vídeos y una pequeña comparativa sobre los softwares de edición de vídeo más populares.

2.1. Formación de una Imagen Digital

Para entender cómo se pueden manipular una imagen o un vídeo y cómo esta manipulación podría ser detectada, es necesario primero conocer el proceso de formación de dichos vídeos o imágenes. En los siguientes apartados se explican las fases por las que pasa la imagen, los elementos involucrados. Finalmente, se hace una pequeña introducción al concepto de ruido que introducen los dispositivos usados para generar las imágenes y vídeos.

2.1.1. Pipeline de una Cámara Digital

El proceso por el cual se obtiene una imagen se conoce como **pipeline**. Este proceso comienza con la captura de la escena que se desea plasmar en un fichero multimedia (imagen o vídeo). A partir de este momento, intervienen los distintos componentes de una cámara hasta tener la imagen en formato digital. Cabe destacar que, a pesar de que el principio es el mismo para todas las cámaras, la mayoría de los detalles más concretos de este proceso dependen del fabricante y,

a menudo, suelen mantenerse confidenciales [1]. En la Figura 2.1 se muestra un esquema de todos los elementos del pipeline por los que pasa la luz hasta ser transformada en una imagen digital.

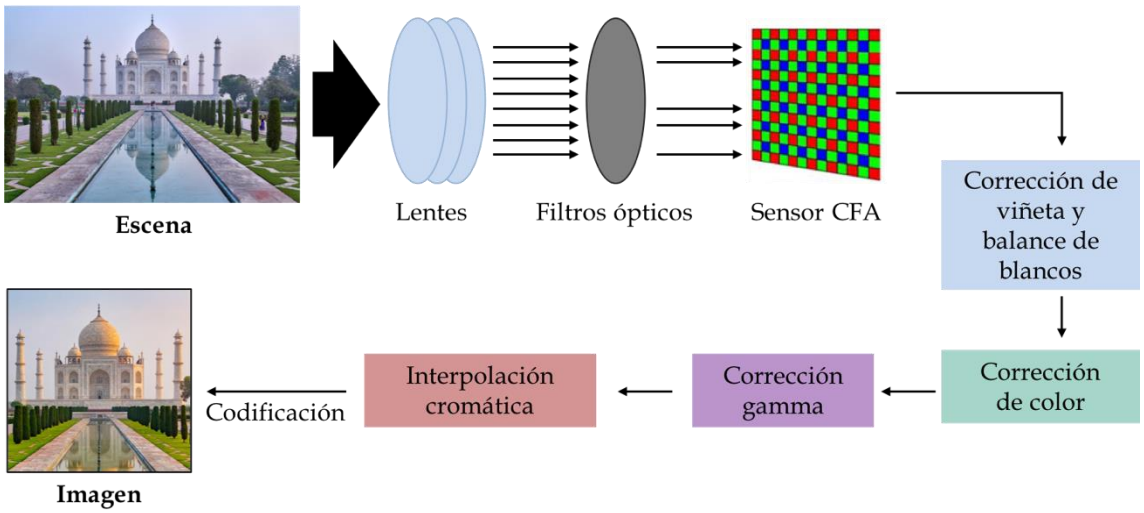


Fig. 2.1: Pipeline o proceso de formación de imágenes digitales.

Cuando se captura una escena, los rayos de luz atraviesan las lentes, que controlan la exposición, la estabilización y el foco de la imagen. Después, esta luz pasa a los filtros ópticos, que solo dejan pasar aquellos que cumplan ciertas propiedades, atenuando o suprimiendo el resto y mejorando la calidad visual de la imagen.

Los rayos restantes llegan al sensor CFA, una matriz de pequeños fotosensores sensibles a la luz que captura la intensidad recibida y la transforma en una señal analógica. Este sensor está unido a un mosaico formado por filtros de color que separan la luz captada en varias bandas de color; normalmente la roja, la verde y la azul.

A continuación, se aplican varias correcciones de forma consecutiva: Primero se ajusta el balance de blancos para una reproducción más fiel, evitando que haya colores dominantes. Después, se corrige el viñetado, un efecto que hace que el brillo o la saturación de la imagen se reduzcan en los bordes en comparación con el centro. Más adelante, se corrigen los colores captados y finalmente, se aplica

una transformación no lineal llamada corrección gamma, que se encarga de compensar varias propiedades para adaptar la imagen a la forma en que el ser humano percibe la luz y el color.

El siguiente paso es la interpolación cromática, un proceso por el cuál la imagen se reconstruye a partir de las muestras cromáticas que fueron captadas por el sensor. Además, en ocasiones, dichas muestras están incompletas, por lo que se estiman a partir de la información conocida y se suaviza la imagen para eliminan algunas impurezas.

Finalmente, se realiza la codificación de la imagen, un proceso por el cual la información de color de cada píxel se pasa a formato digital. Para ello, se almacena en forma de una matriz con distintas bandas de color que el dispositivo en cuestión almacena en memoria y que es capaz de representar para visualizar la imagen.

2.1.2. Ruido de la Imagen

Cada uno de estos elementos por los que la escena capturada pasa durante el pipeline va dejando una huella única. Esta huella terminará por manifestarse en forma de pequeños defectos en la imagen o vídeo que almacena en los dispositivos[2].

A pesar de que durante el procesamiento se incluyen algunos mecanismos para intentar mitigar estas imperfecciones, nunca pueden llegar a ser eliminadas en su totalidad. Esta marca final se denomina **ruido** y es lo que dará pie al análisis forense y permitirá determinar si las evidencias han sido o no manipuladas. El ruido final de una imagen está formado por el **ruido de disparo**, que es aleatorio y distinto en cada toma; y el patrón de **ruido espacial**, que no varía entre capturas. Este patrón de ruido espacial está compuesto las señales de ruido presentadas en la Figura 2.2.

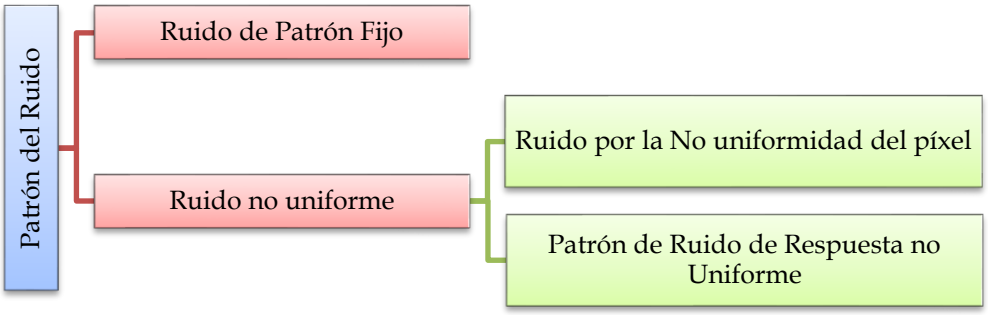


Fig. 2.2: Clasificación del patrón de ruido de una imagen.

- El **ruido de patrón fijo** (FPN) se genera debido a la corriente de oscuridad [3], aunque también intervienen otros factores como la exposición o la temperatura. Este tipo de ruido es independiente de la señal y se añade a los demás, por lo que muchas cámaras son capaces de eliminarlo automáticamente restando un marco oscuro a las imágenes generadas.

- El **ruido por la no uniformidad del píxel** (PNU) es causado por una diferencia de sensibilidad entre los píxeles del sensor. Este fenómeno aparece por imperfecciones inherentes a la fabricación del sensor, como los píxeles muertos o los píxeles calientes. Estos defectos son fruto de la no homogeneidad de las obleas de silicio que componen el sensor.

- El **ruido de respuesta no uniforme** (PRNU) es la parte más dominante del patrón de ruido final y se debe a una diferencia de respuesta de cada píxel a la señal captada. Esta señal de ruido está compuesta por el ruido PNU y por otros defectos presentes en la captura de la imagen como, por ejemplo, la configuración del zoom o las partículas de polvo en las lentes.

Además, aunque los conceptos fundamentales y el procesamiento de la imagen son muy similares en dispositivos móviles y cámaras digitales tradicionales; existen diferencias en factores como la resolución, la apertura de la lente, la distancia focal, la unión de fotogramas, etc. Por este motivo y como se muestra en [4], algunos métodos de detección de manipulaciones pueden resultar más eficaces en evidencias tomadas con unos dispositivos u otros. Más

concretamente, aquellas técnicas que utilicen el ruido del sensor serán más efectivas para imágenes y vídeos capturadas con dispositivos móviles.

2.2. Compresión de un Vídeo Digital

La formación de un vídeo digital mediante la concatenación de todas las imágenes capturadas con el proceso anterior para crear la secuencia de movimiento. Como consecuencia de la gran cantidad de fotogramas, los dispositivos necesitan reducir su tamaño para poder almacenarlo. Para ello, se lleva a cabo un proceso denominado **compresión** que consiste en codificar la misma información de la que se dispone, pero utilizando menos bits.

Aunque el concepto de compresión no es exclusivo de los vídeos digitales, sino que se puede aplicar a cualquier tipo de información cuyo tamaño se quiera reducir; este apartado solo explicará su aplicación en vídeos. En este ámbito, la información que se desea comprimir es, por tanto, todos y cada uno de los fotogramas que conforman la secuencia del vídeo.

Dado que un vídeo es un conjunto de imágenes reproducidas de manera sucesiva; es inevitable que estas contengan una cantidad abundante de información repetida, puesto que su contenido varía muy poco entre fotogramas [5]. Esta información redundante es prescindible y es, por tanto, lo que los algoritmos de compresión intentan eliminar para reducir el tamaño del vídeo. Esta eliminación da lugar a tres tipos de fotogramas:

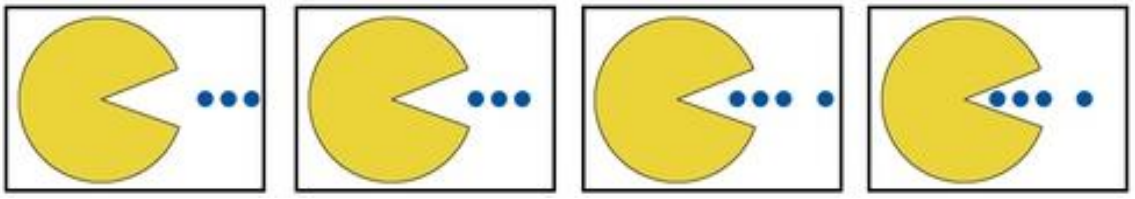
- **Fotogramas-I:** También conocidos como 'intra-coded picture' o fotogramas clave. Son aquellos que contienen una imagen completa y están codificadas sin ninguna referencia a otros fotogramas. Típicamente son los que mayor número de bits requieren para ser representados.

- **Fotogramas-P**: También conocidos como 'predicted-picture' o fotogramas predichos. Son aquellos que contienen únicamente la información de los

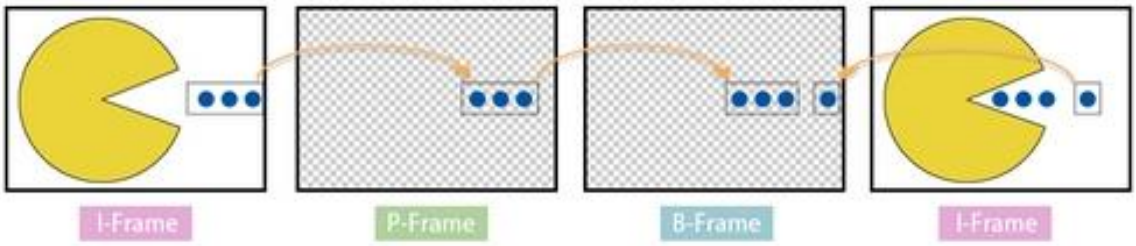
cambios con respecto al fotograma anterior, al que referencian. Requieren aún menos bits que los fotogramas clave para ser almacenados, por lo que ocupan menos memoria.

- **Fotogramas-B**: También se conocen como 'bi-directional predicted picture' o fotogramas predichos bidireccionalmente. Son aquellos fotogramas que contienen la información de los cambios respecto al fotograma anterior y al siguiente. De esta forma, ocupan aún menos que los dos tipos anteriores, pero necesitan conocer los dos fotogramas a los que referencian para poder ser decodificados.

En la Figura 2.3 se muestra un ejemplo sencillo de compresión de la secuencia original en fotogramas I, P y B, para reducir el tamaño de almacenamiento.



(a) Secuencia original



(b) Tipos de fotogramas

Fig. 2.3: Compresión en fotogramas I, P y B.

Como se observa en la figura, se ha reducido el tamaño de la secuencia original (Figura 2.2.a) al eliminar información de algunos de sus fotogramas. El segundo fotograma se puede convertir en un fotograma-P y omitir información que ya contiene el fotograma clave anterior. De manera similar, el tercer fotograma se convierte en un fotograma-B. eliminando aún más información, ya que esta se puede obtener del fotograma anterior y el siguiente, a los que se referencia.

2.2.1. Doble-Compresión de un Vídeo

La compresión de un vídeo tiene como consecuencia inevitable que, si se desea aplicar una modificación en sus fotogramas, será necesario descomprimirlo previamente y volver a comprimirlo después [6]. A este proceso se le denomina **doble-compresión** o **recompresión.** La detección de este fenómeno puede ser de gran utilidad a la hora de identificar si un vídeo ha sido o no falsificado debido a que, aunque no se encuentre la modificación principal, la propia presencia de trazas residuales de una doble-compresión puede delatar la existencia de otra manipulación previa.

2.3. Clasificación de las Manipulaciones en Vídeos

En esta sección se realiza un recorrido por las diferentes técnicas de manipulación para vídeos. Como ya se ha explicado previamente, se entiende un **vídeo digital** como un conjunto de imágenes llamadas **fotogramas** que se reproducen de manera consecutiva en una secuencia. Este concepto, da lugar a dos ámbitos diferentes que permiten clasificar las manipulaciones en dos tipos, según qué dimensión se vea alterada: inter- e intra- fotograma. Es preciso señalar que ambos tipos de manipulación podrían combinarse a la hora de alterar un vídeo.

2.3.1. Manipulaciones Intra-Fotograma

Las manipulaciones intra-fotograma son aquellas que afectan a las imágenes que forman el vídeo de manera individual pero no como conjunto. Es decir, se aplican modificaciones en el contenido de los fotogramas, pero no se altera para nada la línea temporal del vídeo. Es posible clasificar las manipulaciones de este tipo según el nivel de detalle al que se aplican:

- **A nivel de fotograma:** Consiste en tratar el fotograma como una imagen formada y no como el conjunto de sus píxeles. Algunos ejemplos son la

redimensión, la rotación o el recorte.

- **A nivel de píxel:** Consiste en tratar el fotograma como un conjunto de píxeles individuales. Los principales ejemplos de manipulaciones a nivel de píxel son el copia-pega, el empalme, el repintado, el remuestreo y la aplicación de filtros.

Las principales técnicas de manipulación intra-fotograma se explican a continuación, acompañadas de las imágenes de ejemplo.

2.3.1.1. Copia-Pega

Se entiende como 'Copy-Move', 'Copy-Paste', clonación o Copia-Pega [7] aquella manipulación de imágenes digitales que consiste en remplazar contenidos de la imagen original por otros de otra parte de esta. El procedimiento consistiría en seleccionar una zona del fotograma en cuestión, copiarlo y pegarlo justo encima de la sección que se desea manipular. Además, es muy común que a los fragmentos 'pegados' se les apliquen rotaciones, reescalados, filtros de color u otras transformaciones para perfeccionar su integración en la imagen. Todo esto, combinado con técnicas de 'blurring' o emborronado permite realizar un suavizado de los bordes de la región modificada, haciendo más difícil la detección de la manipulación.

Esta manipulación se puede utilizar con dos propósitos principales: o bien para duplicar elementos; o bien para "hacerlos desaparecer" al cubrirlos con una copia de otra parte de la misma imagen. Para esta segunda función son ideales las áreas y fondos con patrones o texturas irregulares, como podrían ser la hierba, el asfalto, la madera, etc. ya que no son fácilmente diferenciables por el ojo humano. En la Figura 2.4 se puede observar una aplicación de la técnica Copia-Pega, en la que se han duplicado los gatos para simular que son más de dos en realidad.

(a) Imagen original (b) Imagen manipulada

Fig. 2.4: Copia-Pega de duplicación.

En cambio, en la Figura 2.5 se ha utilizado el Copia-Pega para ocultar el camión con vegetación, haciendo ver que solamente hay un vehículo.

(a) Imagen original (b) Imagen manipulada

Fig. 2.5: Copia-Pega de eliminación.

2.3.1.2. Empalme

El 'Splicing' o empalme [8] es una técnica de manipulación de imágenes muy similar al Copia-Pega, con la diferencia de que en este caso se los fragmentos que se van a copiar se obtienen de una imagen distinta a la original, a la que llamamos "imagen donante". El objetivo de esta técnica suele ser insertar elementos nuevos en el fotograma. Aunque, al igual que en el Copia-Pega, la sección pegada puede haberse sometido a diversas transformaciones; en este caso puede ser delatada más fácilmente si la imagen donante fue capturada con un dispositivo distinto ya

que entonces, la huella que dejó su propio pipeline es también será distinta. En la Figura 2.6, se muestra un ejemplo de empalme.

(a) Imagen original

(b) Imagen donante

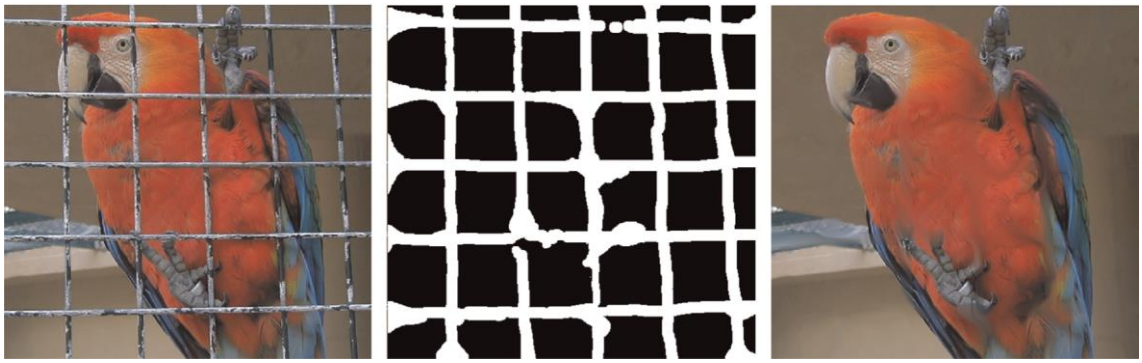
(c) Imagen manipulada (a+b)

Fig. 2.6: Empalme de dos imágenes.

2.3.1.3. Repintado

El 'Impainting' o repintado [8] es una técnica de manipulación y restauración de imágenes que permite rellenar huecos en el fotograma. Se utiliza para recuperar o remover algunas partes de la imagen, pero sin ninguna pérdida perceptual. El concepto es también muy similar al Copia-Pega, pero se puede categorizar como una manipulación distinta ya que en este caso no se pretende copiar un área completa de la imagen sino pequeños parches de distintas partes de esta. Estos parches han de asemejarse mucho a la zona que se desea rellenar y se suelen escoger mediante reglas o algoritmos de aprendizaje automático.

A diferencia de las anteriores manipulaciones, al tratarse de regiones especialmente pequeñas los defectos y el ruido de la imagen pueden ser enmascarados, lo que dificulta su detección. Por este motivo, los algoritmos de detección de manipulaciones Copia-Pega no suelen dar buenos resultados ante manipulaciones de repintado. En la Figura 2.7 se muestra un ejemplo en el que se han utilizado técnicas de repintado para eliminar la rejilla de la jaula de un loro.



(a) Imagen original (b) Máscara binaria (c) Repintado

Fig. 2.7: Repintado con máscara binaria.

2.3.1.4. Remuestreo

El 'Resampling' o remuestreo es una técnica de manipulación que consiste en cambiar el número total de pixeles de una imagen. Suele utilizarse en conjunto con alguna de las manipulaciones anteriores, ya que es común tener que reajustar las dimensiones de las áreas manipuladas. Mientras que una operación de reescalado estiraría los píxeles existentes hasta llegar a las dimensiones deseadas; aplicando técnicas de remuestreo esto se lograría añadiendo los pixeles necesarios, dotando así de una mayor calidad y elasticidad a la imagen, como se muestra en la Figura 2.8.

(a) Imagen original

(b) Resultado del reescalado

(c) Resultado del remuestreo

Fig. 2.8: Reescalado y remuestreo de una imagen.

2.3.1.5. Aplicación de Filtros

El 'Filtering' o aplicación de filtros es una técnica de manipulación que consiste en mejorar o modificar una imagen mediante la alteración, enfatización o eliminación de algunas de sus características. Los distintos filtros pueden ser categorizados en lineales si el valor de un píxel puede ser obtenido mediante la combinación lineal de los píxeles vecinos; o no lineales si esto no fuese posible.

Algunos filtros pueden tener como único fin mejorar el acabado de una imagen modificando su brillo, contraste, color, etc. mientras que otros pueden tener otras aplicaciones como suavizar o eliminar el ruido de una imagen. Aunque la aplicación de filtros por sí misma no tiene por qué ser malintencionada; es importante estudiarla dado que suele utilizarse en

combinación con otras manipulaciones y podría afectar al rendimiento de los algoritmos de detección. En la Figura 2.9 se muestra un ejemplo de la aplicación de un filtro para la eliminación de ruido sobre la imagen de Lena.

(a) Imagen original (b) Imagen manipulada

Fig. 2.9: Aplicación de filtro de eliminación de ruido.

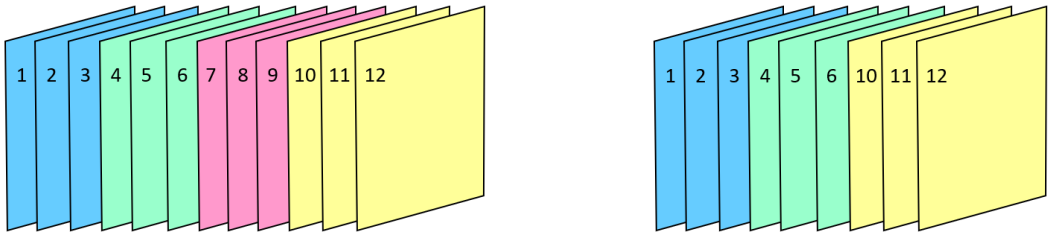
2.3.2. Manipulaciones Inter-Fotograma

A diferencia de las anteriores, las manipulaciones inter-fotograma son aquellas que afectan a la dimensión temporal del vídeo, modificando el flujo de este, pero no las imágenes que lo forman. Simplificando al máximo, las operaciones que se pueden realizar a nivel inter-fotograma son añadir o quitar fotogramas. No obstante, los diferentes usos y combinaciones de estas operaciones pueden dar lugar a distintas técnicas de manipulación que se explican a continuación, acompañadas de imágenes de una secuencia de duración determinada.

2.3.2.1. Eliminación de Fotogramas

Se entiende como eliminación de fotogramas aquella manipulación que consiste en retirar fotogramas de la secuencia del vídeo. Al deshacerse de fotogramas, se consigue que desaparezcan también los hechos que ocurren en ellos, por lo que esta técnica es realmente potente a la hora de llevar a cabo una manipulación.

Si se aplica esta técnica al principio o el final de la línea temporal, se consigue lo que se conoce habitualmente como recortar el vídeo. En cambio, si los fotogramas eliminados se encuentran en el medio de la secuencia, el resultado será un salto entre dos fotogramas que antes no estaban conectados (ver Figura 2.10).

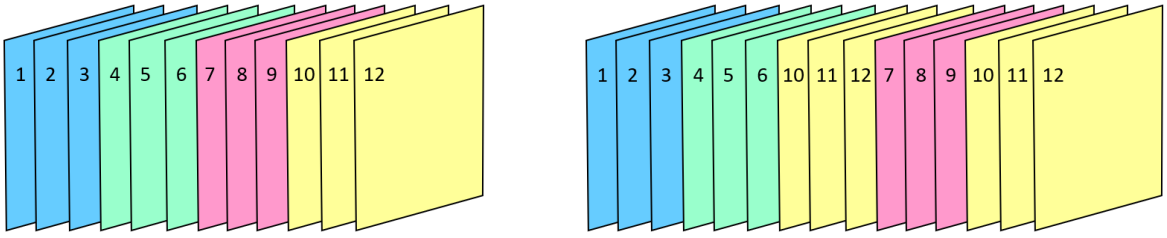


(a) Secuencia original (b) Secuencia manipulada

Fig. 2.10: Eliminación de fotogramas.

2.3.2.2. Duplicación de Fotogramas

La duplicación de fotogramas es una técnica de manipulación de vídeos que consiste en insertar fotogramas que ya existen en el vídeo en otra parte de la secuencia. De este modo, se puede lograr que un elemento permanezca más tiempo en escena o que aparezca repetidas veces. En muchos trabajos esta técnica se conoce también como Copia-Pega inter-fotograma, por su símil con la técnica explicada en la sección anterior, ya que en este caso también se están copiando y pegando partes del vídeo. La diferencia es que, mientras que antes toda la manipulación tenía lugar dentro de los fotogramas, lo que ahora se duplica son fotogramas completos sobre la línea temporal del vídeo (ver Figura 2.11).

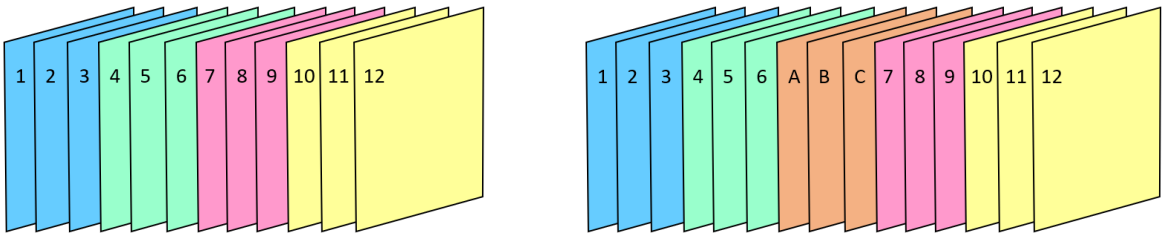


(a) Secuencia original (b) Secuencia manipulada

Fig. 2.11: Duplicación de fotogramas.

2.3.2.3. Inserción de Fotogramas

La inserción de fotogramas es una técnica de manipulación por la que, en la línea temporal de un video, se añaden fotogramas pertenecientes a otro video distinto. El objetivo principal de esta técnica suele ser incriminar elementos que no aparecían inicialmente en el vídeo original. Continuando con el símil con las técnicas intra-fotograma, esta técnica se conoce a veces como empalme inter-fotograma; ya que se están copiando y pegando partes, en este caso fotogramas completos, pertenecientes a la secuencia de otro vídeo diferente (ver Figura 2.12).



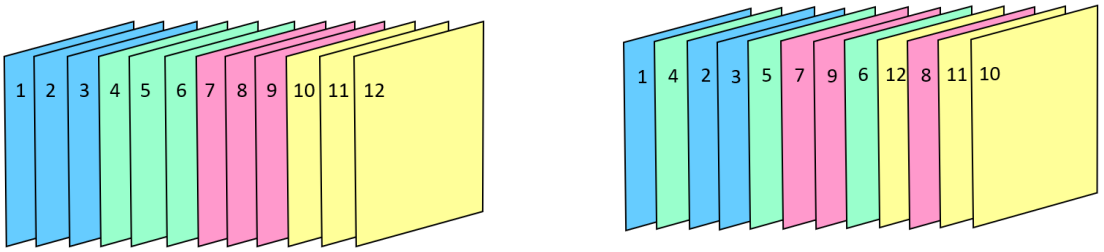
(a) Secuencia original

(b) Secuencia manipulada

Fig. 2.12: Inserción de fotogramas.

2.3.2.4. Mezclado de Fotogramas

El mezclado de fotogramas o 'frame shuffle' es una técnica de manipulación que consiste en cambiar la posición de los fotogramas en la línea temporal del vídeo. Con esto, lo que se consigue es alterar el orden de la secuencia y, por consiguiente, el orden de los hechos que ocurren en ella (ver Figura 2.13).



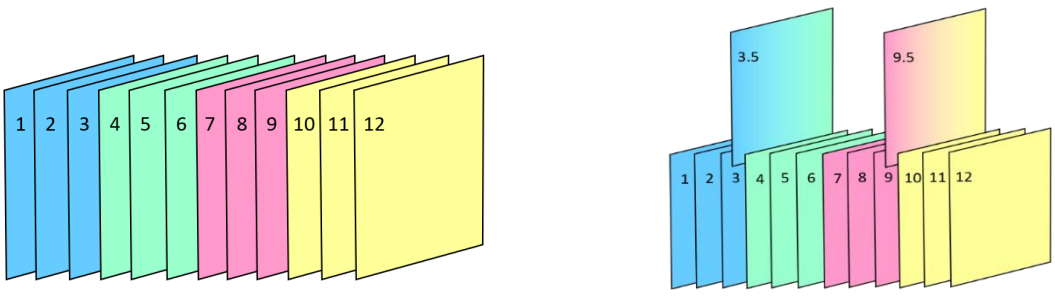
(a) Secuencia original

(b) Secuencia manipulada

Fig. 2.13: Mezclado de fotogramas.

2.3.2.5. Interpolación Temporal entre Fotogramas

Se entiende como interpolación temporal de fotogramas a la manipulación consistente en generar nuevos fotogramas intermedios a partir de los ya existentes. Para ello se utilizan algoritmos que, dadas dos imágenes de la secuencia, son capaces de predecir el movimiento y generar una imagen intermedia a partir de la información de las mismas. Esta manipulación tiene muchas aplicaciones, tales como la compresión de vídeos, la mejora de la visualización o la cámara superlenta, también conocida como 'slow-motion', que nos permite capturar eventos que ocurren a velocidades muy altas (ver Figura 2.14). Normalmente, para aplicar esta técnica y obtener buenos resultados es necesario aplicar también una variación de FPS, que se explica seguidamente.



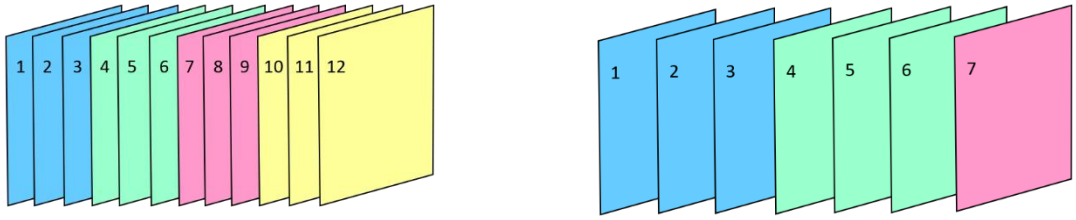
(a) Secuencia original

(b) Secuencia manipulada

Fig. 2.14: Mezclado de fotogramas.

2.3.2.6. Variación de FPS

Se entiende como FPS (fotogramas por segundo) al número medio de imágenes fijas que se muestran en un segundo de reproducción y crean la sensación de movimiento de un vídeo. Cuanto mayor es el valor de los FPS, el cambio entre fotogramas se produce más rápido y se logra una mayor fluidez en el vídeo. La variación de FPS es una técnica de manipulación que consiste en aumentar o reducir el número de fotogramas por segundo del vídeo. De este modo y, a veces en combinación con otras técnicas, se consigue modificar la velocidad de reproducción del vídeo. Otra aplicación es el 'time-lapse', que consiste en capturar fotogramas en intervalos de tiempo durante un periodo muy prolongado; y reproducirlos después con un elevado número de FPS. Con esta técnica se capturan sucesos que acontecen a velocidades muy lentas, como amaneceres, construcciones de edificios, crecimientos de plantas, etc. (ver Figura 2.15).



(a) Secuencia original

(b) Secuencia manipulada

Fig. 2.15: Mezclado de fotogramas.

2.4. Herramientas de Edición de Vídeo

Aunque algunas manipulaciones pueden llegar a tener nivel de profesionalidad muy elevado, otras veces se realizan con herramientas comunes de edición de vídeo. Con el objetivo de entender mejor qué se puede llegar a hacer con este tipo de editores se han analizado y comparado las características que ofrecen algunos de los más utilizados. La Tabla 2.1 muestra una tabla comparativa.

Herramienta	Licencia	Público	Principales Funcionalidades
Filmora	Precio medio Ilimitada/Mensual Tiene versión gratuita	Amateur	Edición sencilla de clips y audio Picture in picture Títulos, créditos y subtítulos avanzados Filtros y transiciones predefinidos Croma
Sony Vegas Pro	Precio muy elevado. Ilimitada/Mensual	Profesional	Edición muy avanzada de clips y audio Picture in picture Zoom-in, zoom-out Filtros y transiciones avanzados Ajustes de color Títulos, créditos y subtítulos avanzados Croma Soporte vídeos 3D y 360 Edición multi-pantalla Corrección de objetivo y estabilizador
Camtasia Studio	Precio Elevado. Ilimitada	Amateur	Edición sencilla de clips y audio Picture in picture Ajustes de color Zoom-in, zoom-out Filtros y transiciones predefinidos Títulos y créditos predefinidos
Adobe Premiere	Precio Muy elevado. Mensual	Profesional	Edición muy avanzada de clips y audio Picture in picture Zoom-in, zoom-out Filtros y transiciones avanzados Ajustes de color Títulos, créditos y subtítulos avanzados Croma Soporte vídeos 3D y 360 Edición multi-pantalla Corrección de objetivo y estabilizador
Pinnacle Studio	Precio Medio. Ilimitada	Amateur	Edición avanzada de clips y audio Picture in picture Zoom-in, zoom-out Filtros y transiciones predefinidos Soporte vídeos 360 Títulos y créditos avanzados
Windows Movie Maker	Incluido en Windows	Principiante	Edición sencilla de clips y audio Picture in picture Filtros y transiciones predefinidos Títulos y créditos predefinidos
iMovie	Incluido en MAC	Principiante	Edición sencilla de clips y audio Picture in picture Filtros y transiciones predefinidos Títulos y créditos predefinidos

Tabla 2.1: Comparativa de herramientas de edición de vídeo.

3. TÉCNICAS DE DETECCIÓN DE MANIPULACIONES EN IMÁGENES DIGITALES Y VÍDEOS

A lo largo de este capítulo se analiza el Estado del Arte, realizando un recorrido comparativo sobre las principales técnicas de detección de manipulaciones, que han servido como base para el algoritmo desarrollado. Como punto de partida, se pueden clasificar las técnicas de detección de manipulaciones en dos categorías principales:

- **Técnicas de detección activas:** Son aquellas en las que la evidencia digital se somete a un procesamiento en el momento de la creación, como la inserción de una marca de agua, una firma digital u otros mecanismos de autentificación. Ante una posible falsificación tan solo habría que tratar de restaurar dicho mecanismo y, en caso de no ser posible, se podría concluir que ha habido una manipulación. El principal inconveniente es que la mayoría de las veces las evidencias no han sido sometidas a ese pre-procesamiento, por lo que no se pueden utilizar estos métodos.

- **Técnicas de detección pasivas:** Son aquellas que parten de la premisa de que el contenido digital tiene una huella digital inherente y única que deja su marca desde el momento en el que la escena es capturada por la cámara. De esta forma, cualquier posible falsificación alteraría las características de dicha huella y delataría la manipulación que se ha realizado. En la mayoría de los casos es necesario utilizar estas técnicas ya que las evidencias no han sido preparadas para su autentificación previamente.

En la Figura 3.1 se muestra un esquema con la clasificación de las técnicas de detección de manipulaciones sobre las que va a tratar el capítulo.

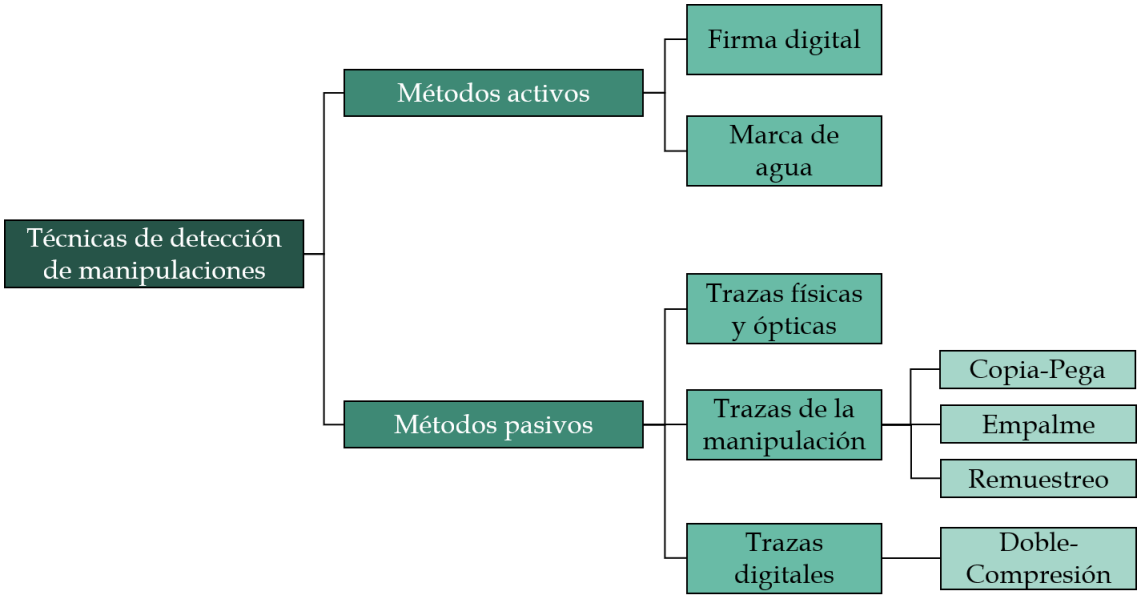


Fig. 3.1: Técnicas de detección de manipulaciones.

3.1. Detección Activa de Manipulaciones

Las técnicas de detección activa de evidencias se basan en introducir previamente a la imagen o vídeo algún mecanismo de autentificación, que delatará cualquier manipulación si resulta alterada a posteriori. Debido a la necesidad de este procesamiento previo también se conocen como técnicas de protección en lugar de detección. Principalmente existen dos técnicas, aunque tienen numerosas variantes e implementaciones. Son la firma digital y la marca de agua.

3.1.1. Firma Digital

El concepto de 'digital signature' o firma digital fue introducido en [9] como un sistema de autentificación que dependiese del contenido de la evidencia en cuestión y, adicionalmente, de otros datos secretos conocidos únicamente por el firmante, como podrían ser los metadatos. La firma digital es, por tanto, una técnica de protección activa de imágenes y vídeos en la cual el valor de los píxeles se introduce a una función de hash y se encripta posteriormente con una clave privada de un cifrado asimétrico.

El principal inconveniente de esta técnica es su rigidez, ya que es muy sensible a cualquier cambio en el valor de los píxeles y en ocasiones un formato de compresión con pérdida o el simple re-guardado pueden alterar e invalidar la firma. Además, esta técnica puede tener vulnerabilidades, como ocurrió con los sistemas de autentificación de imágenes de Canon y Nikon, cuyas claves fueron extraídas por la empresa de seguridad Elcomsoft, que fue capaz de generar imágenes falsas que superaban con éxito la verificación. [10] [11].

Debido a los problemas mencionados, se han implementado variantes como la firma robusta de imágenes, propuesta en [12], que es más flexible a postprocesamientos como la compresión.

3.1.2. Marca de Agua

La marca de agua digital o 'watermarking' es una técnica de protección activa que permite incluir nuevos datos en evidencias digitales. Por ello, es comúnmente utilizada para garantizar derechos de autor o de propiedad de imágenes y vídeos.

En [13] se realiza una revisión de los métodos y algoritmos que se han ido desarrollando a lo largo de los años con esta finalidad. Típicamente se insertan etiquetas o códigos incrustados que identifiquen de forma exclusiva el contenido, permaneciendo en él incluso tras ser sometidos a procesos de transformación.

En [14] se propone una implementación de un código de posición secuenciado aleatorio (RSPPMC) como marca de agua oculta embebida para imágenes JPEG. A la vista de las conclusiones expuestas, parece tener suficiente robustez para sobrevivir a compresiones con pérdida, ajustes de color y a filtrados de paso bajo.

En [15] se presenta otro método de marca de agua en el que, aprovechando las propiedades de la transformada discreta del coseno (DCT), se esconde una

cadena de 50 bits pseudo-aleatorios generados por una función de 'hash' en la imagen. Los resultados demostraron robustez ante manipulaciones como la inserción de ruido, la transformación a escala de grises, la aplicación de filtros, la rotación y el reescalado.

3.2. Detección Pasiva de Manipulaciones

Las técnicas pasivas de detección de manipulaciones se basan en la utilización de trazas y huellas que se han ido dejando sobre el vídeo o imagen de manera no intencionada al someterla a ciertos procesos. Las características que se pueden extraer de estas trazas suelen obedecer a ciertos patrones estadísticos y por tanto las anomalías son fácilmente relacionables con una posible falsificación.

Podemos clasificar este tipo de trazas en tres categorías según su origen:

- **Trazas ópticas y físicas:** Aparecen desde el momento en que se captura la escena. Las luces y sombras, la geometría, el suavizado, la aberración cromática y la distorsión radial de las lentes son algunos ejemplos.

- **Trazas digitales**: Su adquisición se da en el procesamiento y almacenamiento de la escena capturada. Algunos ejemplos son el sensor, los niveles de ruido, la compresión o los metadatos.

- **Trazas de la manipulación**: Aparecen en el momento en que altera el contenido de la imagen o vídeo. Pueden delatar que se ha realizado un cambio malintencionado como un Copia-pega, un empalme, un repintado, un remuestreo o una aplicación de un filtro.
  En este trabajo se pretende estudiar la detección de manipulaciones a partir de trazas de esta categoría, como se explica en los siguientes capítulos.

3.2.1. Detección de Copia-Pega

Para la detección de este tipo de manipulaciones se busca un fragmento de la imagen o un fotograma que haya sido duplicado, según sea Copia-Pega intra- o inter-fotograma. En ambos casos, será posible encontrar similitudes que pueden ser utilizadas para delatar la falsificación.

En [16] se realiza un recorrido sobre algunos métodos de detección de manipulaciones Copia-Pega que es muy útil como introducción. Para cada método se realiza una evaluación que concluye, a pesar de que son más caros computacionalmente hablando, los algoritmos basados en bloques dan mejores resultados y son menos sensibles a transformaciones que los basados en puntos clave.

Uno de los trabajos más tempranos sobre la detección del Copia-Pega en imágenes fue [7], en el que se proponen dos algoritmos que analizan la imagen por bloques en busca de picos de correlación. El primero de los algoritmos propuestos detecta coincidencias exactas en la imagen lo cual, como los propios autores dicen, no resulta demasiado útil por ser un escenario irreal, ya que normalmente las áreas duplicadas son también expuestas a transformaciones. El segundo de los algoritmos busca coincidencias similares utilizando la transformada del coseno (DCT) con resultados convincentes, aunque solo probado con tres imágenes. Aunque no plantea una posible utilización en vídeos, sirvió como fuente para trabajos posteriores para este tipo de manipulaciones.

En [17], la propuesta es detectar manipulaciones de Copia-Pega intrafotográmico en vídeos de escena estática mediante la extracción y análisis de las características del ruido. Para ello, utiliza una función que estima los niveles de ruido y detecta así los valores aislados del mismo en los fotogramas. El método resulta interesante, pero la detección pierde efectividad ante zonas con colores muy brillantes, generando un alto porcentaje de falsos positivos.

En [18] se propone un método capaz de detectar manipulaciones de tipo Copia-Pega, tanto en imágenes independientes como en fotogramas de un vídeo. Para ello, analiza las características de bloques conectados y categorizándolos mediante un algoritmo de pertenencia difusa. El principal inconveniente de este algoritmo es que no funciona con imágenes y vídeos que hayan sido comprimidos.

El método de detección de [19] es capaz de localizar falsificaciones a nivel de bloque en vídeos. Para ello, se divide el vídeo en conjuntos de fotogramas y se buscan anormalidades en la coherencia espacio-temporal en dichos conjuntos, clasificándolos como manipulados. El algoritmo obtiene buenos resultados, aunque empeora significantemente ante vídeos comprimidos.

En [20], el método se basa en el estudio de las características SIFT de sus fotogramas. La propuesta es interesante dado que los resultados muestran robustez a distintas transformaciones geométricas, y capacidad de detectar más de una región alterada en un mismo vídeo.

[6] propone un método para la detección de Copia-Pega en imágenes y en vídeos basándose en la correlación entre sus fotogramas. Dado que el algoritmo para vídeos inicialmente solo obtiene buenos resultados cuando han sido grabados por cámaras estáticas, en este trabajo se propone también una variación que es capaz de detectar la región modificada en vídeos con cámara móvil. El principal problema es que el porcentaje de detección disminuye bastante cuando las regiones manipuladas son pequeñas o han sido sometidas a alguna transformación.

En [21] y en [22] se propone prácticamente el mismo método: un algoritmo basado en la correlación del ruido residual entre fotogramas capaz de detectar manipulaciones Copia-Pega, ya sean de tipo inter- o intra-fotograma. Como primer paso obtiene la versión libre de ruido de los fotogramas utilizando la DWT, y se resta a los fotogramas originales para obtener exclusivamente el

ruido residual de estos. A continuación, se divide el resultado en ventanas que se recorren calculando la correlación cruzada ("cross-correlation") entre cada una de ellas y su equipolente en el siguiente fotograma. Tras esto, se aplica un modelo de mezcla Gaussiana, cuyos parámetros se ajustan con un algoritmo de máxima verosimilitud, para estimar la distribución estadística que debería seguir cada una de las dos manipulaciones. Finalmente, se utiliza un clasificador Bayesiano para determinar a cuál de las dos distribuciones se aproximan más los datos obtenidos, clasificando así la manipulación con una precisión bastante elevada (mayor que 95%) en todos los experimentos.

3.2.2. Detección del Empalme

El empalme en imágenes y vídeos es una de las manipulaciones más potentes, ya que permite, entre otras cosas, introducir en escena cualquier hecho que no haya ocurrido en realidad. Prácticamente todas las técnicas de detección de este tipo de manipulaciones parten de la misma base: el área pegada puede ser delatable mediante la variación en el patrón de las características de la imagen o vídeo originales.

Uno de los primeros trabajos publicados fue [23]. En él se demuestra que cualquier manipulación no-lineal, como es por ejemplo un empalme; introduce variaciones en características como la bicoherencia o la correlación de la imagen, que pueden ser detectadas mediante el análisis de la señal de la imagen.

En [24] se propone un algoritmo que permite detectar manipulaciones de tipo empalme en vídeos. Para ello, se realiza una extracción de características como vectores de movimiento e intensidad del gradiente, que sirven como entrada para un algoritmo de aprendizaje automático basado en una máquina de soporte vectorial. Este tipo de algoritmos tratan de encontrar el mejor hiperplano que divida todas las muestras en dos o más clases diferenciadas, que este caso son

manipulada y no manipulada. Los resultados destacaron en vídeos con un fondo estático.

3.2.3. Detección del Remuestreo

En [25] se presenta una técnica de detección una manipulación de remuestreo a la que denomina recorte-ampliado y que consiste en aumentar el tamaño de un fotograma hasta que una parte quede fuera del marco formado por las dimensiones originales del mismo y sea eliminada. Para ello, aplicaban un filtro MACE-MRH e identificaban la variación que se había producido en las correlaciones estadísticas del contenido, así como en el SPN. Esta propuesta requiere de un complicado ajuste de parámetros, pero obtiene buenos resultados incluso detectando la aplicación de recorte-ampliado en combinación con otras manipulaciones.

3.2.4. Detección de la Doble-Compresión

La doble compresión es una consecuencia inevitable de cualquier otra manipulación. Su detección es de gran importancia ya que, a la hora de analizar un vídeo, puede darse el caso de que la detección de una doble compresión revele una falsificación que no se estaba detectando de otro modo.

En [6] se realiza una de las primeras propuestas para exponer la doble compresión. Los autores demostraron que, al someter un vídeo a una recompresión MPEG, los fotogramas clave se someten también a una recompresión JPEG y que, además, los fotogramas que cambian de grupo de imágenes durante el proceso producen variaciones que medirían después a través de la estimación del movimiento.

Tres años más tarde, los mismos autores publicaron [26], donde propusieron un algoritmo capaz de detectar la doble cuantificación resultante de una doble compresión MPEG con características distintas mediante el análisis de los

coeficientes DCT. Los resultados mostraron que el método propuesto era especialmente eficaz en vídeos que utilizaban un croma o '*green-screen*'. El principal inconveniente es que para que el método funcionase la calidad de la segunda compresión debía haber sido mayor que la de la primera.

[26] utilizaron los conceptos del método anterior y analizaron la distribución estadística que seguían los coeficientes DCT cuantificados en busca de un patrón convexo fruto de la doble compresión. A la vista de los resultados, esta técnica fue eficaz para diferentes sistemas de codificación, aunque su precisión se veía reducida cuando la tasa de bits de la segunda compresión era menos que la primera o cuando los vídeos utilizaban cámara lenta.

4. ALGORITMO PARA LA DETECCIÓN DE COPIA-PEGA INTRA-FOTOGRAMA

Este capítulo se explica cómo se ha llevado a cabo la implementación del algoritmo de detección, utilizando [22] como guía, su funcionamiento y las modificaciones o aportaciones que se han realizado sobre él.

4.1. Conceptos Generales

Para poder comprender algunas de las técnicas utilizadas en el algoritmo es importante conocer ciertos conceptos que se explican en esta sección.

4.1.1. FFMPEG

FFMPEG[27] es una colección de herramientas de software libre y sencillas de utilizar orientadas al procesamiento de vídeos y audio. Algunas de sus funcionalidades principales son la grabación, codificación, decodificación, conversión y reproducción de distintos formatos de vídeo y audio. Fue lanzada en el año 2000 y desarrollada en GNU/Linux, aunque es compatible con otras configuraciones.

La colección está compuesta por diversas componentes y bibliotecas, aunque en este algoritmo se ha utilizado la componente básica, que permite realizar todas las operaciones por la línea de comandos. Más concretamente, se ha utilizado para la extracción de fotogramas clave y para la codificación del vídeo resultante al final del programa.

4.1.2. Transformada Discreta de Wavelet

La Transformada Discreta de Wavelet (DWT) es una herramienta matemática

que permite representar una señal mediante la variación de sus valores, dando lugar a una onda. Este concepto tiene grandes aplicaciones en el campo del procesamiento de imágenes, y uno de ellos es, precisamente, la **eliminación de ruido** o "denoising", como se explica en [26].

Dada una imagen, en este caso el fotograma en cuestión, como señal de entrada; es posible representarla solo con los valores de los contrastes entre sus regiones suaves y sus regiones más abruptas, que serían los bordes. De este modo, al no utilizar el valor de todos y cada uno de los píxeles sino solo la variación de estos, se consigue representar la imagen con un menor número de coeficientes, siendo más sencillo eliminar la información redundante, es decir, el ruido y quedar solo con las características significativas. Una vez hecho esto, tan solo habría que aplicar la Transformada Discreta de Wavelet inversa para obtener la imagen libre de ruido.

4.2. Funcionamiento

El algoritmo desarrollado tiene como objetivo detectar falsificaciones de tipo Copia-Pega intra-fotograma con fondo estático, como las que se podrían realizar sobre vídeos captados por cámaras de seguridad. La entrada y salida del programa es el vídeo que se desea analizar. El diagrama de funcionamiento del algoritmo se presenta en la Figura 4.1.

Como se muestra en la Figura 4.1, el algoritmo necesita como única entrada una carpeta con todos los vídeos que se desea analizar. Esta carpeta es configurable en el código mediante el valor de la constante **dataset_path**, que debe contener la ruta absoluta de la misma.

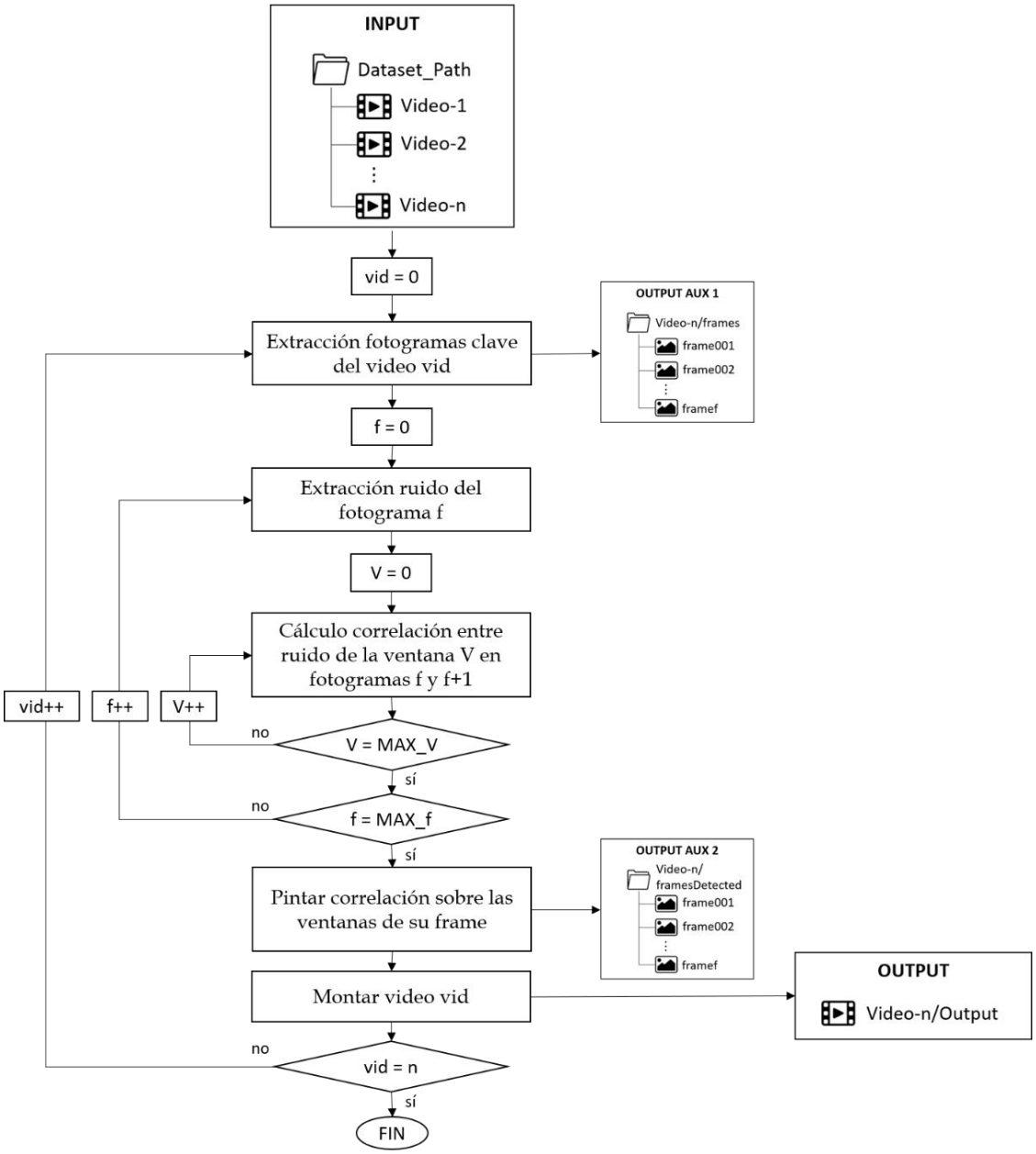


Fig. 4.1: Diagrama de flujo del programa.

El proceso de análisis se repetirá para cada uno de los vídeos de nuestro dataset, guardando los resultados auxiliares y finales en una subcarpeta llamada Video-n (siendo n la cantidad de vídeos de entrada) que se creará automáticamente:

1. Como primer paso se divide el vídeo que se desea analizar en fotogramas. Dado que el algoritmo base inicialmente no profundizaba en la explicación sobre el método de extracción de los fotogramas, se ha decidido utilizar únicamente los fotogramas clave (fotogramas-I) porque, como se explicó en el Capítulo 2.2, son los que más información pueden proporcionar para el posterior análisis. Para llevar a cabo esta extracción se llama a la función **extract_type_frames**, que, utilizando la librería FFMPEG, extrae los fotogramas del tipo elegido en formato .PNG y al 100% de calidadad, dejándolos en la carpeta deseada, que en esta implementación será una subcarpeta llamada frames.

2. A continuación, los fotogramas extraídos son transformados a escala de grises, dado que no altera el funcionamiento del algoritmo. Con esto se reduce la representación del color en RGB a una única banda en lugar de tres, facilitando el manejo de los datos y haciendo un menor uso de memoria.

3. Una vez hecho esto se desea obtener el ruido de cada uno de ellos, ya que es con lo que se va a trabajar y no con toda la información de la imagen. Esto se lleva a cabo utilizando la función **dwt_denoise**, que obtiene el ruido aplicando un filtro de eliminación de ruido basado en la DWT con el que se obtiene la imagen en bruto, libre de ruido. Llegados a este punto, es trivial obtener exclusivamente el ruido (R) realizando la resta entre el fotograma original (F) y el fotograma (F') libre de ruido.

$$R = F - F'$$

4. Seguidamente, se recorre la imagen en bloques de tamaño N x N a los que denominaremos **ventanas** (siendo N configurable en el código mediante la constante WINDOW_SIZE). Esta es una técnica comúnmente utilizada en algoritmos de este tipo ya que, al analizar la imagen en fragmentos más pequeños, la manipulación queda acotada en una sección y es más sencillo detectarla y localizarla.

Para cada ventana, se calcula el coeficiente de correlación entre ella misma y su ventana equipolente en el fotograma siguiente. Esto se repite para todos los fotogramas. Para esta operación se ha utilizado la función **numpy.corrcoef,** de la librería NUMPY**.** En la Figura 4.2 se muestra un ejemplo del cálculo de la correlación para una ventana V.

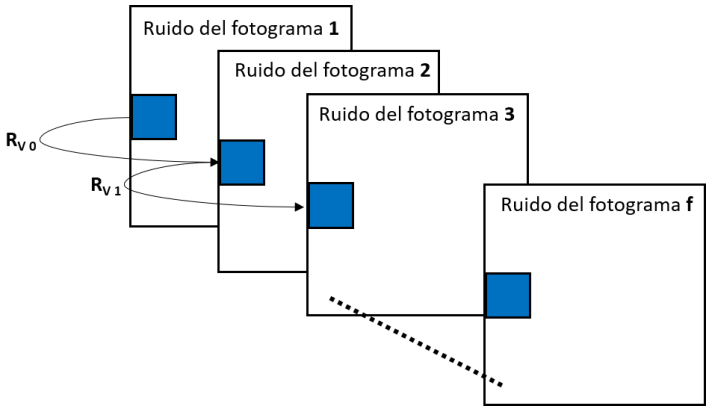


Fig. 4.2: Cálculo de la correlación por ventanas.

Con el objetivo de reducir la complejidad de los cálculos y, en consecuencia, el tiempo de ejecución del programa, manteniendo la capacidad de detectar manipulaciones de tipo Copia-Pega intra-fotograma en los vídeos analizados; se ha utilizado la fórmula de la correlación simple en lugar de la correlación cruzada. La correlación simple es un cálculo estadístico más sencillo cuyo resultado es un coeficiente que toma valores situados entre 0 y 1. Se calcula de la siguiente manera:

$$R = \frac{\sum_i \sum_j (n_{i,j}^{t} - \overline{n}^{\,t})(n_{i,j}^{t-1} - \overline{n}^{\,t-1})}{\sqrt{\sum_i \sum_j (n_{i,j}^{t} - \overline{n}^{\,t})^2 \sum_i \sum_j (n_{i,j}^{t-1} - \overline{n}^{\,t-1})^2}}$$

Donde:

- n es el ruido medio de una ventana
- i, j definen la posición de dicha ventana
- t es el número de fotograma

- $\bar{n}$ es la media ponderada del ruido del fotograma, que se define como:

$$\bar{n}^{\,t} = \frac{\sum_i \sum_j n_{i,j}^t}{V^2}$$

- V el número total de ventanas.

Este coeficiente R indica el grado en que dos variables están relacionadas, siendo mayor a medida que se acerca a 1. Aplicando esta definición a nuestra implementación, el valor de correlación indicaría cómo de parecido es el ruido de la ventana evaluada a lo largo de los fotogramas. Dicho de otro modo, una ventana con una correlación muy elevada sería sospechosa de haber sido manipulada mediante un Copia-Pega porque mantiene prácticamente el mismo ruido a lo largo de la secuencia, lo cual no puede ocurrir salvo que el video haya sido alterado.

Una vez terminado el proceso de cálculo, el resultado es un vector de coeficientes de correlación (R).Dichos vectores, a su vez, se van adjuntando en una matriz de resultados; de forma que en las filas estarán cada una de las ventanas (V) y en las columnas cada uno de los frames (f). Esta matriz acabará teniendo un aspecto similar al mostrado en la Figura 4.3.

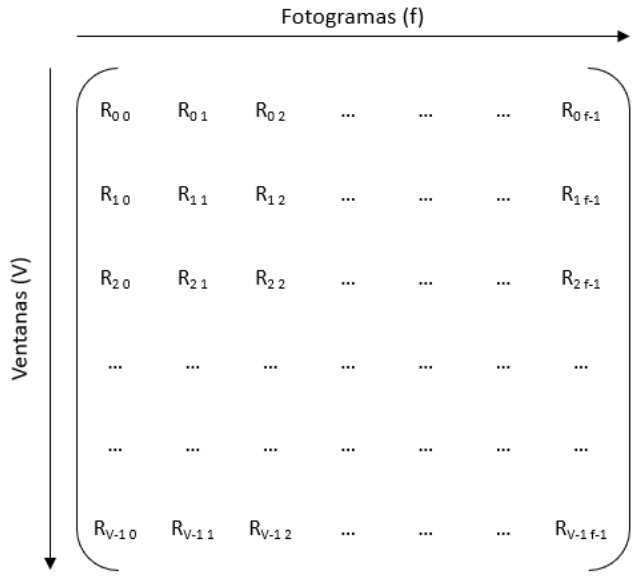


Fig. 4.3: Matriz resultados.

5. El siguiente paso es, en cada fotograma, colorear las ventanas según los coeficientes de correlación que se han calculado. Para ello, se obtiene el color que corresponderá a la ventana multiplicando su coeficiente de correlación por 255, valor que representa el blanco en un píxel con una sola banda de color (escala de grises). Esta operación se realiza en la función **paintResults**.

$$\mathrm{P(V, f)} = 255 \cdot R_{V\,f}$$

De este modo, una zona sospechosa de haber sido manipulada quedaría coloreada prácticamente en blanco y se mostraría bien delimitada; mientras que las ventanas originales del resto del fotograma solamente presentarían ruido. Además, así no es necesaria la implementación de una clasificación para distinguir las zonas modificadas de las originales, pues las manipulaciones se distinguen a simple vista al reproducir el vídeo resultante. El motivo de esta implementación es que, para que una clasificación sea óptima es necesario que sus umbrales también lo sean, lo cual suele conseguirse al calcularlos mediante algoritmos capaces de realizar una

predicción, que necesitan ser entrenados previamente con diferentes conjuntos de datos. Este tipo de algoritmos de aprendizaje automático realizan cálculos internos muy complejos, lo que ralentiza la ejecución del algoritmo.

6. Finalmente, en una carpeta llamada framesDetected se guardan los fotogramas sobre los que se ha coloreado en el paso anterior y se utilizan para montar el vídeo final, que se guardará con el nombre de Output.avi. Para codificar el vídeo resultante a partir de los fotogramas de nuevo se hace uso de la herramienta **FFMPEG.**

En la Figura 4.4 se muestra un ejemplo del uso del algoritmo completo. En la imagen central (b) se puede observar como el hombre que estaba sentado en el banco de la imagen original (a) ha sido eliminado y cómo esta manipulación queda expuesta en la detección (c) tras aplicar el método explicado anteriormente.

(a) Fotograma original (b) Fotograma manipulado (c) Detección

Fig. 4.4: Detección de fotograma manipulado.

4.3. Herramientas de Desarrollo

El desarrollo del proyecto se ha llevado a cabo utilizando el lenguaje de programación Python en su versión 2.7; y, para la implementación del algoritmo se ha hecho uso de las siguientes librerías:

- **FFMPEG**: Librería que contiene herramientas para el manejo de audio y

vídeos. Entre sus funcionalidades destacan la codificación, la decodificación, la redimensión y la reproducción.

- **NUMPY**: Librería que facilita la realización de cálculos matemáticos como, por ejemplo, el cálculo del coeficiente de correlación. Además, incluye tipos que hacen más sencillo el almacenamiento de vectores y matrices.

- **OPENCV**: Librería destinada al análisis de imágenes y aprendizaje automático que incluye funciones que facilitan el uso y tratamiento de imágenes y vídeos.

- **PILLOW**: Librería que permite la apertura, manipulación y escritura de distintos formatos de imagen.

- **PYWAVELETS**: Librería que permite descomponer la imagen en sub-bandas, lo cual tiene, entre otras aplicaciones, la obtención de las características wavelet.

5. EXPERIMENTOS Y RESULTADOS

En este capítulo se describen las pruebas que se han realizado con el fin de evaluar la eficacia y resultados del algoritmo desarrollado ante distintos tipos de manipulaciones Copia-Pega y distintos formatos de vídeo. A continuación, en la Tabla 5.1, se muestran las características del equipo en el que se han realizado todos los experimentos.

Recurso	Características
Sistema Operativo	Ubuntu 16.04
Memoria RAM	4 GB
Procesador	Intel® Celeron® CPU N2840 @ 2.16GHz
Gráficos	Intel® HD Graphics
Arquitectura	64bits
Disco Duro	500 GB HDD

Tabla 5.1: Características del equipo de pruebas.

5.1. Dataset Utilizados

Para evaluar el rendimiento del algoritmo en cuanto a la detección de manipulaciones Copia-Pega intra-fotograma, se han utilizado un total de 10 vídeos sobre los que se ha realizado este tipo de manipulación.

- La mitad de los vídeos pertenecen al conjunto de datos público SULFA[29], que incluye, entre otros, vídeos grabados con cámaras estacionarias de distintos modelos sobre los que posteriormente se aplicó una manipulación Copia-Pega intra-fotograma.

- El resto de los vídeos han sido grabados con la cámara de un dispositivo móvil Xiaomi MI A1 colocado sobre un trípode. Posteriormente han sido manipulados utilizando funciones del módulo Image de la librería PILLOW para duplicar regiones de los fotogramas.

Las características de todos estos vídeos se muestran en la Tabla 5.2.

Dataset	Cámara	Num vídeos	Formato	Códec	Resolución	Duración media
SULFA [29]	FUJIFILM 2800	4	.AVI	Uncompressed 8-bit RGB	320x240	9.75s
	Canon SX2000	1	.MOV	H.264	320x240	12s
Propio	Xiaomi MI A1	5	.mp4	Uncompressed 8-bit RGB	1280x720	8s

Tabla 5.2: Características del dataset utilizado.

5.2. Tamaño Óptimo de la Ventana

Durante el proceso de análisis de un vídeo, este se divide en ventanas cuadradas cuyo tamaño es configurable mediante el valor de la variable WINDOW_SIZE, que determina el número de píxeles que mide cada lado. Por ejemplo, si WINDOW_SIZE=8, se analizará la correlación del ruido en ventanas de 8x8 píxeles.

Para descubrir el tamaño de ventana óptimo se realizaron pruebas con tres versiones de distintas resoluciones de un mismo vídeo manipulado. Las dimensiones utilizadas fueron 360x240, 720x480 y 1080x720 (HD) por ser las más comunes. La Tabla 5.3 contiene los resultados de este experimento evaluando, para cada resolución y tamaño de ventana, la sencillez con la que la región manipulada se diferencia a simple vista tras el pintado de los fotogramas manipulados.

Resolución	WINDOW_SIZE	Tiempo de ejecución (s)	Detección zona manipulada
360x240	8	92	Muy diferenciada
	16	33	Diferenciada
	32	19	Diferenciada
720x480	8	412	Muy diferenciada
	16	151	Muy diferenciada
	32	81	Diferenciada
1080x720	8	1008	Muy diferenciada
	16	362	Muy diferenciada
	32	224	Muy diferenciada

Tabla 5.3: Comparativa Tamaños de Ventana.

Aunque el nivel de diferenciación en la zona manipulada detectada es mayor cuanto menor es el tamaño de ventana escogido; a la vista de los resultados se puede concluir que la diferencia no se ve tan recompensada en cuanto al tiempo de ejecución. Por este motivo, para garantizar la relación entre el tiempo de ejecución y el resultado obtenido, 16 sería un tamaño de ventana óptimo para los tamaños de vídeos contemplados y, eventualmente, se podría utilizar un tamaño de 8 para vídeos más pequeños.

5.3. Resistencia al Reescalado

En los resultados del experimento anterior, se pudo observar como el algoritmo es resistente a las operaciones de reescalado. Volviendo a la tabla 5.3, queda reflejado como la detección en un vídeo de gran tamaño y en uno de pequeño tamaño dan igualmente buenos resultados y, sin embargo, el tiempo de ejecución es infinitamente menor para el de dimensiones más pequeñas.

Por este motivo y como se explica a continuación, durante la experimentación con vídeos de dimensiones muy grandes, se ha aplicado primero una operación de reescalado para reducirlos hasta 360x240 y así obtener una ejecución más rápida del algoritmo.

Para realizar estas operaciones de reescalado se ha utilizado la opción "scale" de la herramienta FFMPEG, pero manteniendo siempre el resto de las características de los vídeos originales para evitar pérdidas de información que pudiesen repercutir de algún modo a los resultados.

5.4. Detección de Copia-Pega

Tras aplicar el algoritmo a los 10 vídeos de los dataset utilizados, se observa que la detección tiene éxito en los 9 vídeos que no han perdido información durante

la compresión (códec "Uncompressed 8-bit RGB"). En la Figura 5.1 se muestra el resultado obtenido para un vídeo grabado por una cámara de tráfico en el que se ha eliminado uno de los coches. Como se puede ver, la región de carretera que se ha duplicado para tapar el coche aparece claramente diferenciada y pintada de blanco; mientras que, una vez el coche ha salido de escena y comienzan a reproducirse los fotogramas originales, las correlaciones vuelven a mostrar un patrón natural.

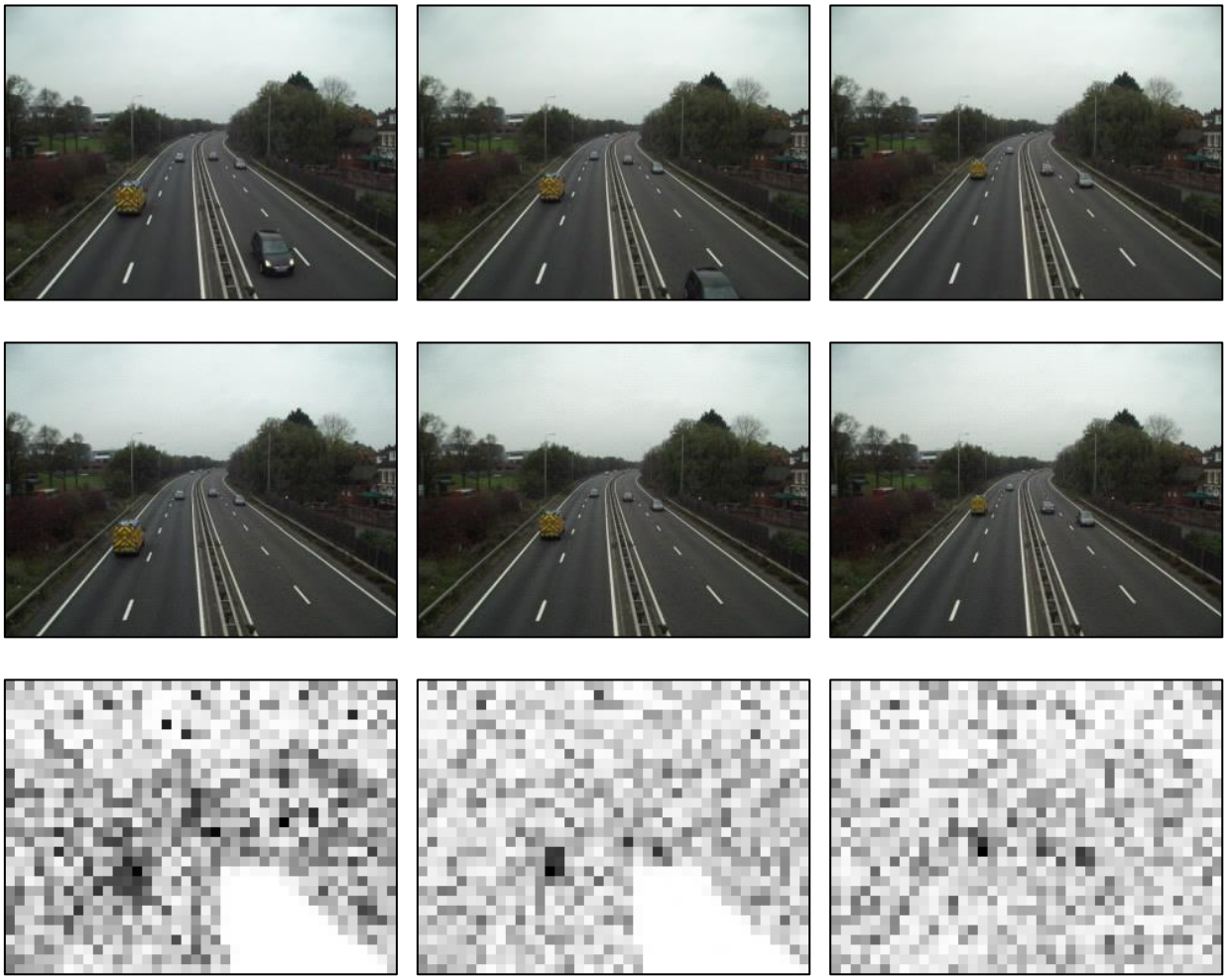


Fig. 5.1: Ejemplo de detección 1.

En la Figura 5.2 se muestra otro ejemplo exitoso de este mismo experimento. En él, una cámara de seguridad graba como dos personas pasan por la escena caminando, pero una de ellas ha sido eliminada ocultándola bajo una región de

pared que ha sido duplicada. De nuevo, al pintar las correlaciones, la zona manipulada que oculta a la persona en cuestión queda claramente diferenciada en color blanco mientras que el resto del fotograma presenta un patrón de correlaciones normal.

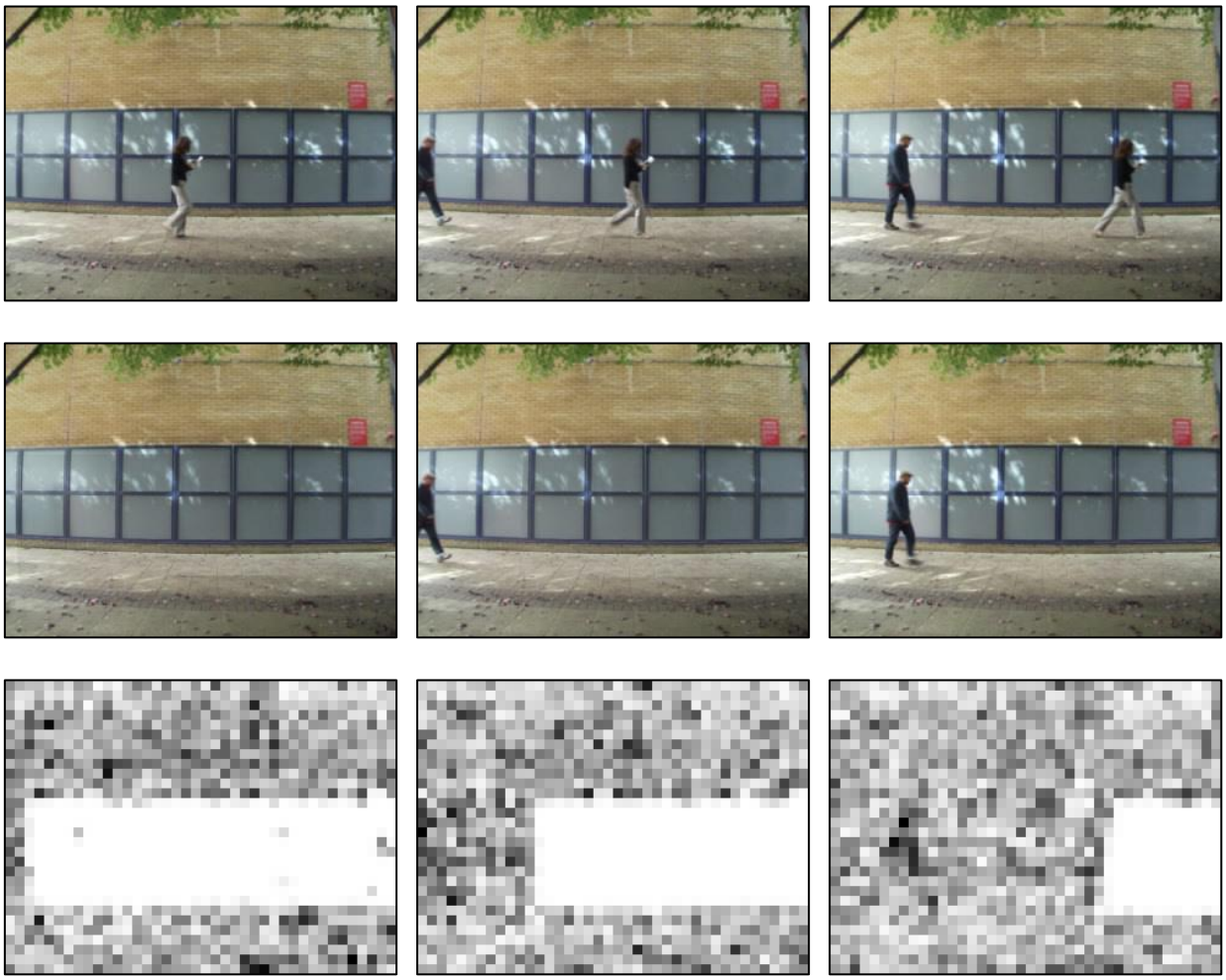


Fig. 5.2: Ejemplo de detección 2.

En la Figura 5.3 se muestra un último ejemplo exitoso de este mismo experimento. Esta vez el vídeo pertenece al dataset propio. En él, se observa como una pelota de baloncesto es lanzada rondando a lo largo de un pasillo. En la versión manipulada la región del suelo vacío de un fotograma ha sido duplicada para ocultar la escena, pero esta se evidencia como una región blanca diferenciada en el resultado de la detección.

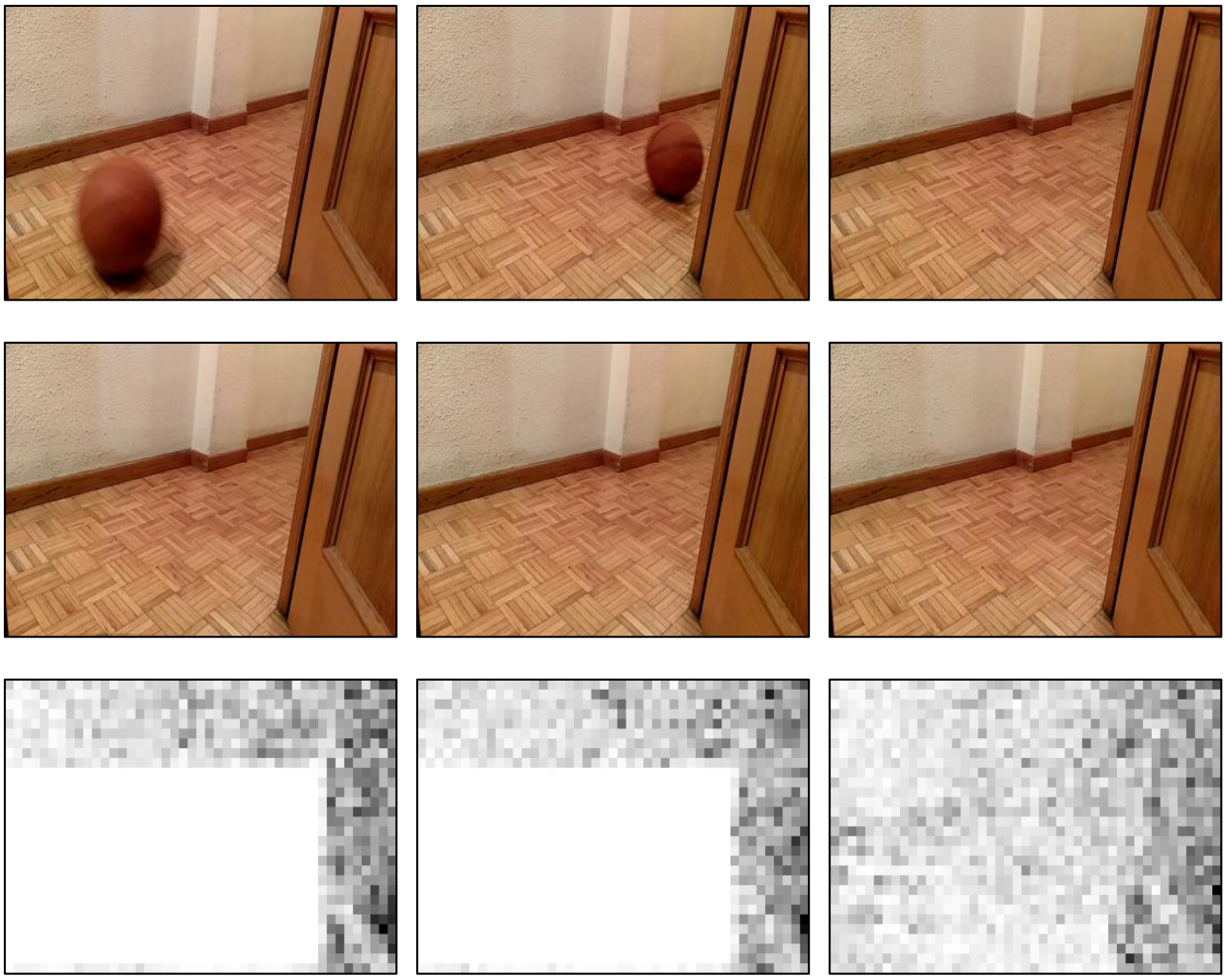


Fig. 5.3: Ejemplo de detección 3.

Dado que el método no obtuvo buen resultado para el vídeo que había sido comprimido siguiendo el códec H.264, se intentaron diferentes transformaciones, pero la pérdida de información causada por la compresión ya había distorsionado el por lo que, como se observa en la Figura 5.4, no fue posible detectar la manipulación.

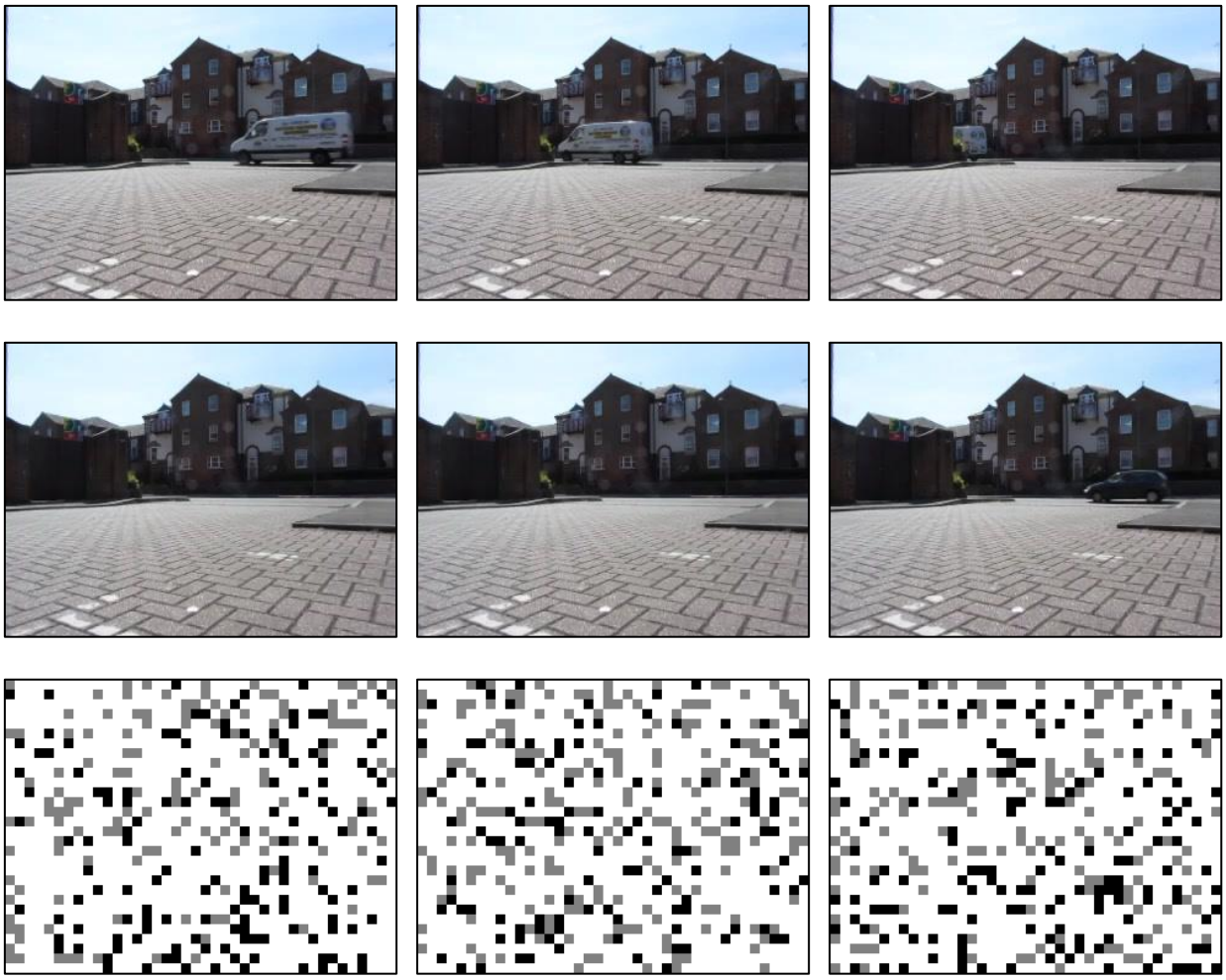


Fig. 5.4: Ejemplo de detección 4.

Para completar el experimento, se ejecutó el programa también sobre los vídeos originales (antes de manipular) y se comprobó que todos presentaban patrones de correlación naturales; es decir, no había ningún caso de falsos positivos.

Para contabilizar los resultados se han seguido los mismos indicadores que se proponen en [22], que son los siguientes:

$$Exactitud\ (E) = \frac{N_{(TP)} + N_{(TN)}}{N_{(TP)} + N_{(TN)} + N_{(FP)} + N_{(FN)}}$$

$$Precisión\ (P) = \frac{N_{(TP)}}{N_{(TP)} + N_{(FP)}}$$

$$Reclamación\ (R) = \frac{N_{(TP)}}{N_{(TP)} + N_{(FN)}}$$

Donde:

- $N_{true\ positive\ (TP)}$ es el número de fotogramas manipulados que fueron detectados como tal.
- $N_{true\ negative\ (TN)}$ es el número de fotogramas auténticos que fueron detectados como tal.
- $N_{false\ positive\ (FP)}$ es el número de fotogramas manipulados que fueron detectados como auténticos.
- $N_{false\ negative\ (FN)}$ es el número de fotogramas auténticos que fueron detectados como manipulados.

En la Tabla 5.4 se muestra una comparativa del método propuesto en este trabajo y el propuesto en [22].

Indicador de rendimiento	[22]	Propuesta propia
Exactitud	97.36-100%	97.49-100%
Precisión	97.34-100%	97.28-100%
Reclamación	99.80-100%	99.55-100%
Tiempo de ejecución	--	44.20s
Resistente a recompresión	Sí	No

Tabla 5.4: Resumen de resultados de la detección.

Tras la realización de estos experimentos y el resumen anterior, se puede observar que el algoritmo propuesto obtiene unos resultados muy similares al algoritmo que sirvió como base en la detección de manipulaciones Copia-Pega intra-fotograma. Con el algoritmo propuesto se ha conseguido una exactitud algo mayor, pero la precisión y la reclamación son levemente menores. No obstante, aunque las diferencias son prácticamente insignificantes, las mediciones serían mucho más representativas si los datasets de ambos trabajos fueran más grandes.

Además, dado que en la implementación desarrollada el análisis de correlación se realiza de manera diferente, se ha perdido la resistencia a

recompresiones, pero se ha logrado un tiempo de ejecución medio muy razonable para un equipo de bajas prestaciones como el que se ha utilizado para la realización de los experimentos. Por este motivo, la propuesta podría ser útil como una primera herramienta de detección o de descarte antes de someter el vídeo a otros algoritmos que realizan un análisis más complejo y, por consiguiente, tienen un tiempo de ejecución mucho mayor.

5.5. Sensibilidad a la Compresión

Con motivo de la detección fallida en los vídeos que habían sido comprimidos utilizando el códec H.264, se procedió a realizar un nuevo experimento. En él, tres de los vídeos cuya manipulación sí había sido detectada eran recomprimidos, pasando del códec "Uncompressed 8-bit RGB" a "H.264", y analizados nuevamente.

Como resultado, los tres vídeos obtuvieron un patrón de ruido ilegible y deteriorado, sobre el que no se pudo detectar ninguna manipulación. De esta manera, se concluye que el algoritmo no es válido para vídeos comprimidos con pérdida de información y, por tanto, es un aspecto que se podría mejorar en un futuro.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. Conclusiones

En primer lugar, se realizó una introducción con la que se ha buscado ofrecer una imagen completa de todos los conceptos necesarios para entender el campo del análisis forense de videos e imágenes, así como un estudio de las técnicas de manipulación más relevantes.

En segundo lugar, tomando como base lo expuesto en el recorrido por el Estado del Arte, se analizó un algoritmo en concreto para la detección de manipulaciones Copia-Pega, que serviría como línea de guía a la implementación del algoritmo propio.

Después, se estudió la posibilidad de introducir modificaciones en el algoritmo explicado para intentar optimizarlo, ya fuese simplificándolo, mejorando sus resultados o reduciendo su tiempo de ejecución. Esto dio lugar a la implementación de cambios en el método de extracción de fotogramas, el cálculo de la correlación y el pintado de los resultados.

Finalmente, se dio paso a una fase experimental en la que se realizaban pruebas sobre el algoritmo utilizando vídeos tanto de un conjunto de datos público como generados para este trabajo. Se realizaron un conjunto de pruebas con diferentes datasets.

En primer lugar, se realizó un experimento para determinar el tamaño óptimo de las ventanas en las que el algoritmo divide los fotogramas para su procesamiento y análisis. Se observó que los resultados de la detección no varían de manera significativa pero que el tiempo de ejecución del programa se reducía considerablemente al aumentar dicho tamaño, siendo 16x16 píxeles el tamaño ideal.

A la vista de los resultados del experimento anterior y realizando pruebas con otros vídeos, se observó que el algoritmo desarrollado es resistente a las operaciones de reescalado, por lo que se puede reducir el tamaño de los vídeos antes de analizarlos para agilizar la ejecución.

A continuación, se sometieron a análisis los diez vídeos del conjunto de datos, tanto en su versión original como en la manipulada, haciendo un total de veinte muestras. Se observó una detección exitosa para los vídeos que habían sido comprimidos sin pérdida de información, pero fallida para los vídeos que si utilizaban un códec de compresión. Además, el tiempo de ejecución del programa fue razonable y los indicadores de exactitud, precisión y reclamación presentaron valores similares al algoritmo que ha servido como base.

Finalmente, para concluir que el algoritmo realmente era sensible a las compresiones con pérdida de información se recomprimieron algunos vídeos del conjunto de datos y se volvieron a analizar en busca de manipulaciones. Se observó que los patrones de ruido estaban deteriorados y, por tanto, la detección no era posible.

Dados los resultados de los experimentos mencionados anteriormente, la implementación desarrollada obtiene buenos ratios de detección, aunque no es capaz de detectar manipulaciones en vídeos que hayan sido sometidos a una o varias compresiones con pérdida de información, debido al deterioro de las características del ruido.

Por este motivo y a la vista de los resultados obtenidos, se puede concluir que el uso de este algoritmo estaría condicionado por el formato del vídeo que se desea analizar.

6.2. Trabajo Futuro

Es importante saber qué aún quedan caminos a seguir de cara a profundizar en los aspectos que no se terminaron de desarrollar en este proyecto, ya sea porque requerían conocimientos más específicos, o porque estaban fuera del alcance de este trabajo. No obstante, dado que aportarían valor al estudio de las técnicas de detección de manipulaciones, propongo las siguientes líneas de trabajo a futuro:

- Ampliar el algoritmo desarrollado para conseguir la detección de otros tipo de manipulaciones.
- Perfeccionar la implementación para conseguir que el programa sea robusto ante distintas recompresiones de vídeo.
- Estudiar los principales algoritmos de BigData y aprendizaje automático y su posible aplicación para una mejor eficiencia en el método propuesto.
- Generar y publicar un dataset propio para poder ser de utilidad a futuros investigadores, de la misma manera que otros lo fueron para mí.

RESUMEN EN INGLÉS

7. INTRODUCTION

7.1. Motivation

The motivation behind this project is the investigation, implementation and evaluation of an algorithm for detecting forgeries, which are more and more frequent in a society where the flow of information is increasing while its contrast is every time less.

Thus, the final goal of this project is to approach all the resources provided by the university, with the advice of my tutors and my own research work, in order to extend my knowledge about an increasingly relevant field of development. Also, I have sought this goal by applying all the concepts learned along these years to the resolution of problems and the improvement of algorithms, more specifically, of the one used as basis for this project.

For this, a review over the State of Art will be performed to know the current state of research and the chance of introducing new features that improve the efficiency on detection. These results will be shown graphicly and could be served as basis for future investigations related to forensic analysis.

7.2. Context

This End of Degree Paper is part of a research project entitled RAMSES, approved by the European Commission within the Horizon 2020 Research and Innovation Framework Programme (H2020-FCT-2015, Innovation Action, Proposal Number: 700326) and in which the GASS Group of the Software Engineering and Artificial Intelligence Department participates, integrated in the Faculty of Computer Science of the Complutense University of Madrid (Analysis, Security and Systems Group, http://gass.ucm.es, group 910623 of the catalogue of research groups recognised by the UCM).

In addition to the Complutense University of Madrid, the following participate entities:

- Treelogic Telematics and Rational Logic for Empresa Europea SL (Spain)
- Ministério da Justiça (Portugal)
- University of Kent (United Kingdom)
- Centro Ricerche e Studi su Sicurezza e Criminalità (Italy)
- Fachhochschule fur Offentliche Verwaltung und Rechtspflege in Bayern (Germany)
- Trilateral Research \& Consulting LLP (United Kingdom)
- Politecnico di Milano (Italy)
- Service Public Federal Interieur (Belgium)
- Universitaet des Saarlandes (Germany)
- Dirección General de Policía - Ministerio del Interior (Spain)

7.3. Objectives

This Project has the following objectives:

- To obtain a knowledge basis that allows the complete understanding of forensic analysis and all concepts related to digital videos such as capture, encoding, compression, denoising, etc.

- To study the current ltierature in order to classify and summarize the main forgery techniques, explaining the most important aspects of each of them.

- To study the State of Art of video forgery detection techniques with emphasis on the ones that detect Copy-Move manipulations.

- To design and implement an algorithm capable of detecting intra-frame Copy-Move forgerys over static background, such as those that could be done

over security cameras recorded videos.

- To evaluate the results of the proposed method using different datasets.
- To collaborate on the research of forensic analysis for digital images and videos, serving as a source and reference for future investigations.

7.4. Work Plan

The development of this TFG was divided in three core phases for greater organization:

1. **Reasearch/Investigation Phase**: This phase was the most extensive due to the fact that it involved understanding the context in which the work was going to be carried out, analyzing the current state of related research and finally finding the weaknesses of the techniques and proposing possible solutions for them. In this Phase, the following activities were carried out:
    - The first phase of the project consists mainly of reading articles, studies and scientific publications, with the objective of obtaining all possible information on the subject before addressing it. Even though forensic analysis in computer science has attracted me on several occasions, I had never considered developing something that had to do with it and therefore, the initial knowledge level for this TFG was very basic. I studied, among others, documents on the hardware components of a camera, the process of generating images and videos since they are captured on a mobile device, video compression methods, common motivations and uses of image altering, different manipulations that can be performed on these formats, possible measures to detect or prevent these manipulations, etc.
    - Next, I dove deeper into video and image manipulation techniques, classifying them into different categories and observing and collecting

data sets from each of them. In parallel, I studied and compared algorithms and proposals for the detection of the aforementioned manipulations, understanding their mathematical and statistical foundations; until choosing one whose complexity, utility and technology are consistent with the scope of this TFG.

- At this point, the algorithms that would serve as the basis for the design of the one to be developed were selected, based on those that had better results analyzing the noise pattern left by the camera sensor in video frames.

2. **Execution Phase:** This phase is composed of the following stages: design, implementation and testing. In it, the development environment was prepared and the knowledge acquired during the software development and preparation of test data was applied. The activities developed in this stage are the following:

   - Based on the previous investigations, an algorithm was designed and implemented following the guidance of the selected research, documenting the process and the resources used at all times. During this phase there were some complications with the chosen algorithm and adjustments had to be made in the investigation to solve it. Sometimes returning to the previous phase was necessary in order to study alternative ways to solve the problems that had been found.

   - Once the implementation was completed, tests were carried out with various datasets used in other related investigations, collecting them from the Internet., gathered from the Internet. It is here that the algorithm was gradually refined identifying its weak points. Various modifications that would improve the detection rate and scalability of the algorithm were studied.

- After the improvements to the algorithm were implemented, another set of experiments was carried out to compare the result.

3. **Documentation and Follow-up Phase**: This phase was developed in parallel with the other two phases. Everything done in the implementation of the TFG was documented. The research work was monitored and controlled in Phase 1, as was the technical support in the development of Phase 2. The following activities were carried out:

    - Weekly work coordination meetings were held with project managers. Initially the meetings were twice a week and in them the work was monitored, and they also included workshops for the acquisition of research skills such as: research methodologies, appropriate bibliographic searching for research, use and compilation of documents in LaTex, tools for the management of bibliographic references, free software tools for data analysis such as Python, statistical techniques for signal processing, among others.

    - Everything done in the project was digitally documented in order to prepare the memory of the TFG in parallel. Once the algorithm tests were completed, conclusions were drawn from the experiments carried out and future work activities were proposed that can be developed to improve the developed algorithm and continue advancing the investigation.

7.5. Work Structure

The rest of the memory is organized in 5 more chapters with the structure discussed below.

Chapter 2 gives an introduction to the concepts related to forensic analysis of digital images and videos, as they are necessary for the full understanding of this

work. The process of forming digital images will be briefly explained, from the moment they are captured by the camera of the mobile device until they are used to create the sequence of the video that will be compressed and stored. In addition, the most common video manipulation techniques will be explained and classified, accompanied by graphic examples.

Chapter 3 presents a State of the Art of the main techniques for detecting the different types of forgeries, already discussed in the previous chapter, as well as the advantages and disadvantages of each of the analyzed algorithms. Emphasis will be placed on the method that has motivated and served as the basis for developing the proposed technique..

Chapter 4 describes in detail the Copy-Move detection algorithm developed, explaining the work that has been done and all the tools that have been used in order to achieve the result.

Chapter 5 shows the experiments performed to evaluate the efficiency of the proposed algorithm. Also, the datasets that have been used in the tests are detailed here and an analysis and comparison of the results obtained in each of them is carried out..

Finally, Chapter 6 sets out the main conclusions of this work and a brief reflection of what could be the lines of future research work.

8. CONCLUSIONS AND FUTURE WORK

8.1. Conclusions

In the first place, the present work tries to provide a complete picture of all the concepts that are necessary to understand the field of forensic analysis of images and videos, and a study of the most relevant manipulation techniques as well.

In the second place, taking as a base what was exposed in the State of the Art, a specific algorithm for the detection of Copy-Move forgeries was analyzed, which would serve as a guideline for the implementation of the proposed algorithm.

Afterwards, the possibility of introducing some variations in the explained algorithm was studied in order to try to optimize it, either by simplifying it, improving its results or reducing its computing time. This resulted in the implementation of changes affecting the frame extraction, the calculation of the correlation and the impainting of results.

Finally, there was an experimental phase where the algorithm was tested using videos from a public dataset and videos generated for this work. The following tests were done.

First, an experiment was conducted to determine the optimal size of the windows into which the algorithm divides the frames for its processing and analysis. It was observed that the results of the detection do not vary significantly but the execution time was reduced considerably when increasing this size, being 16x16 pixels the ideal size.

In view of the results of the previous experiment and by testing with other videos, it was found that the algorithm developed is robust to reescalation operations, so the size of the videos can be reduced before analyzing them to speed up the execution.

Next, the ten videos of the dataset were analyzed both in their original and manipulated versions, for a total of twenty samples. Successful detection was observed for videos that were compressed without loss of information, but failed for videos that did use a compression codec. In addition, the execution time of the program was reasonable and the indicators of accuracy, precision and recall presented similar values to the algorithm that has served as a basis.

Finally, to conclude that the algorithm was actually sensitive to lossy compressions, some videos from the dataset were recompressed and reanalyzed for manipulation. It was observed that the noise patterns were deteriorated and therefore detection was not possible.

Given the results of the experiments mentioned above, the implementation developed obtains good detection ratios, although it is not capable of detecting manipulations in videos that have been subjected to one or several lossy compressions, due to the deterioration of the characteristics of the noise.

For this reason and in view of the results obtained, it can be concluded that the use of this algorithm would be conditioned by the format of the analyzed video.

8.2. Future Work

It is important to know that there is still a long ways to go in order to expand in the unfinished aspects of this project, either because they required a more specific knowledge, or because they were out of scope in this project. However, due to they would bring added-value to the the research of forgery detection techniques, I propose the following future lines of work:

- Extend the developed algorithm so it becomes capable of detecting other types of forgeries
- Improve the implementation in order to make the program robust to different video compressions.

- Study main Big-Data and Machine Learning Algorithms and their posible application to the proposed method for a better performance.
- Generate and publish an own dataset in order to be useful for future researchers in the same way they were for me.

9. BIBLIOGRAFÍA

[1] T. H. Thai, F. Retraint, and R. Cogranne, "Camera model identification based on the generalized noise model in natural images," *Digit. Signal Process. A Rev. J.*, vol. 48, pp. 285–297, 2016.

[2] M. Jahanirad, A. W. A. Wahab, and N. B. Anuar, "An evolution of image source camera attribution approaches," *Forensic Sci. Int.*, vol. 262, pp. 242–275, 2016.

[3] J. Nakamura, *Image sensors and signal processing for digital still cameras*. 2017.

[4] O. Çeliktutan, B. Sankur, and I. Avcibaş, "Blind identification of source cell-phone model," *IEEE Trans. Inf. Forensics Secur.*, vol. 3, no. 3, pp. 553–566, 2008.

[5] K. Sitara and B. M. Mehtre, "Digital video tampering detection: An overview of passive techniques," *Digit. Investig.*, vol. 18, pp. 8–22, 2016.

[6] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double quantization," *MMandSec'09 - Proc. 11th ACM Multimed. Secur. Work.*, pp. 39–47, 2009.

[7] J. Fridrich, D. Soukal, and J. Lukáš, "Detection of Copy-Move Forgery in Digital Images," *Int. J.*, vol. 3, no. 2, pp. 652–663, 2003.

[8] M. A. Qureshi and M. Deriche, "A bibliography of pixel-based blind image forgery detection techniques," *Signal Process. Image Commun.*, vol. 39, pp. 46–74, 2015.

[9] M. E. Hellman and D. Whitfield, "New Directions in Cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[10] D. Sklyarov, "Forging Canon Original Decision Data," p. 33, 2010.

[11] V. Katalov, "Nikon Image Authentication System: Compromised," p. 2011, 2011.

[12] X. S. Hua, X. Chen, and H. J. Zhang, "Robust video signature based on ordinal measure," *Proc. - Int. Conf. Image Process. ICIP*, vol. 1, pp. 685–688, 2004.

[13] I. J. Cox, M. L. Miller, and I. Way, "A review of watermarking and the importance of perceptual modeling," vol. 3016, pp. 92–99.

[14] E. Koch and J. Zhao, "Towards robust and hidden image copyright labeling," *IEEE Work. Nonlinear Signal Image …*, pp. 20–23, 1995.

[15] J. Fridrich and M. Goljan, "Robust hash functions for digital watermarking," *Proc. - Int. Conf. Inf. Technol. Coding Comput. ITCC 2000*, pp. 178–183, 2000.

[16] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 6, pp. 1841–1854, 2012.

[17] M. Kobayashi, T. Okabe, and Y. Sato, "Detecting video forgeries based on noise characteristics," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5414 LNCS, pp. 306–317, 2009.

[18] S. Das, G. Darsan, S. L, and D. Devan, "Blind Detection Method for Video Inpainting Forgery," *Int. J. Comput. Appl.*, vol. 60, no. 11, pp. 33–37, 2012.

[19] C. S. Lin and J. J. Tsay, "A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis," *Digit. Investig.*, vol. 11, no. 2, pp. 120–140, 2014.

[20] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A SIFT-based forensic method for copy-move attack detection and transformation recovery," *IEEE Trans. Inf. Forensics Secur.*, vol. 6, no. 3 PART 2, pp. 1099–1110, 2011.

[21] C. C. Hsu, T. Y. Hung, C. W. Lin, and C. T. Hsu, "Video forgery detection using correlation of noise residue," *Proc. 2008 IEEE 10th Work. Multimed. Signal Process. MMSP 2008*, pp. 170–174, 2008.

[22] R. C. Pandey, S. K. Singh, and K. K. Shukla, "A passive forensic method for video: Exposing dynamic object removal and frame duplication in the digital video using sensor noise features," *J. Intell. Fuzzy Syst.*, vol. 32, no. 5, pp. 3339–3353, 2017.

[23] H. Farid, "Detecting Digital Forgeries Using Bispectral Analysis," *Mit Ai Memo Aim-1657, Mit*, pp. 1–9, 1999.

[24] C. Richao, Y. Gaobo, and Z. Ningbo, "Detection of object-based manipulation by the statistical features of object contour," *Forensic Sci. Int.*, vol. 236, pp. 164–169, 2014.

[25] D. K. Hyun, S. J. Ryu, H. Y. Lee, and H. K. Lee, "Detection of upscale-crop and partial manipulation in surveillance video based on sensor pattern noise," *Sensors (Switzerland)*, vol. 13, no. 9, pp. 12605–12631, 2013.

[26] Y. Su, "Detection of Double-compression in MPEG-2 Videos," no. 1, 2010.

[27] "FFMPEG." [Online]. Available: ffmpeg.org/.

[28] M. Lang, H. Guo, J. E. Odegard, C. S. Burrus, and R. O. Wells, "Noise reduction using an undecimated discrete wavelet transform," *IEEE Signal Process. Lett.*, vol. 3, no. 1, pp. 10–12, 1996.

[29] “Surrey University Library for Forensic Analysis (SULFA).” [Online]. Available: sulfa.cs.surrey.ac.uk/videos.php.